What is a Database?

second edition

Viji Kumar

Copyright © Viji Kumar 2023

Chapter 1 – Explanatory Framework

A database is viewed in the following discussions as an organised collection of data that can be accessed, managed, and updated electronically. It is assumed to be designed to enable the collection and storage of information in a structured and efficient way. The information collected in these databases can then be retrieved and manipulated by digital applications or software, making databases an especially useful tool.

A database consists of tables, i.e., rows and columns, with each column being a specific type of data and each row containing a set of values related to that data. The tables can be linked by relationships, which allow data to be analysed across multiple tables. These linked data are what make databases useful.

Incidentally, it would be prudent to bear in mind that there almost certainly will be data protection legislation in most countries to regulate the collection and use of some types of data, e.g., personal data about individuals. In May 2023 Meta (owner of Facebook) was fined €1.2bn by the EU for privacy violations and ordered to suspend transfers of user data to the US.

Databases provide a way for organisations to access copious amounts of data efficiently, arguably enabling these organisations to make better decisions, improve operations, and deliver better services to their customers. However, the danger of bad actors using the data collected for nefarious purposes, e.g., manipulation of an electorate, is an ever-present threat.

Our changing world, especially changes brought about by human activity, necessitates data having to be processed continuously to enable us to be aware of how our environment is evolving. Collecting this data is vital to help formulate policies for the efficient functioning of, among other things, the economy, health and social care and other infrastructure elements of an advanced 21st century nation.

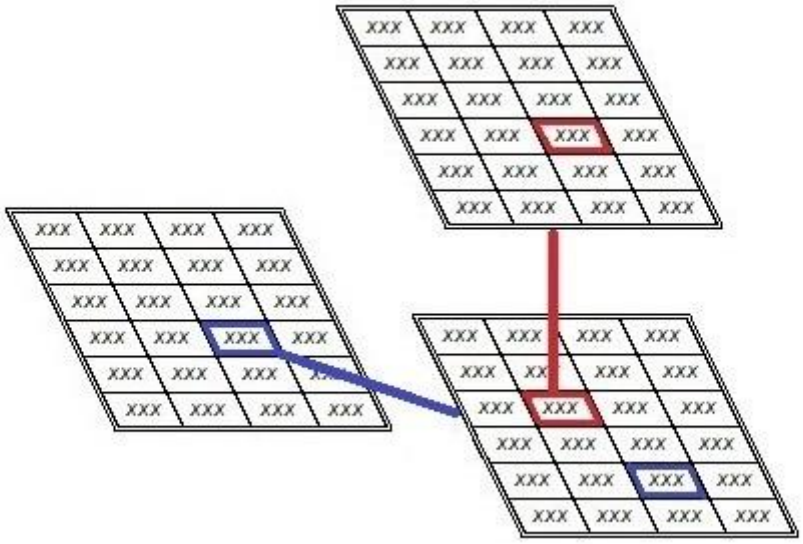
The aim of this book is to enable the reader to review data that they may come across in their daily lives e.g., the results of clinical trials of a vaccine and be able to evaluate the usefulness and the provenance of the data being provided. One of the reasons data such as epidemiological statistics are referenced by any organisation or authority is to justify policies that they are advocating or implementing. Knowing a bit about structured information may be helpful when reviewing the data underpinning such policy justifications. Well-constructed databases allow users to search substantial amounts of data in an efficient fashion to obtain useful insights.

This book is for non-database specialists e.g., students who wish to appreciate some of the factors that must be considered when searching and analysing information presented as tables of data or the visual representations of such data e.g., graphs and charts. Additionally, there will be comments about whether there is any need to probe into the provenance of the data being analysed or reviewed.

team name
falcons
eagles
dolphins
sharks

A list of football teams.

Data are defined as discrete items of information e.g., names of individuals or organisations, dates when events occurred, performance measurements and other metrics. The simplest collection of data is a list e.g., of names of individuals or products. These data items can be linked to other discrete items of information, e.g., addresses, or amounts invoiced producing collections of linked data or databases. The ***function of a database,*** for the purposes of this explanation, is to enable the collection of searchable structured data about related entities e.g., sporting teams and their performances over time in different competitions or musicians and their albums and/or songs.



Here, the ***form of a database*** is viewed as a collection of linked tables. Much of the value of a database arises from the relationships that are implemented by the linkages between its component tables. The rows of these tables are populated by strings of characters. Characters may be letters *(of any alphabet)*, numbers *(i.e., Arabic numerals and zero)*, spaces, punctuation marks and other symbols. The value of each ***data item*** collected and recorded in a database table is an ***attribute*** of an ***entity*** that is of interest to the data collection exercise. e.g., the passport number of an individual crossing a border.

Entities are defined, in this explanation, as constructs that are distinct and with a separate existence e.g., an individual, a credit card or a football team. Entities can also refer to separate, definable, and distinct constructs that have no material existence e.g., league seasons, families, or codons. Examples of data items, i.e., the attributes of diverse types of entities, include an individual's or a team’s name, the time or place of a transaction conducted using a debit or credit card, the venue of a

game of football or the RNA bases in a codon. Some attributes, e.g., unique identifiers *(IDs)* and names, are common to many distinct types of entities, e.g., individuals, objects, and organizations.

Names, identifiers, events, and measurements are “raw data” often collected to enable other information to be derived for the benefit of the users of a database, e.g., the total number of games won by a team during a league season. To enable the collected data to be effectively processed, it is necessary to classify the data items collected. The ***datatype*** is the categorisation used to classify data items in the tables. Data management systems require this categorisation to be able to perform the appropriate operations on the data, as a data item is classified according to its contents e.g., whether the value recorded is text or a number, a number with a decimal component or an integer, or if the data are of specific or esoteric formats such as dates or digital links. The three datatypes featured in this book will be extremely mundane, text strings (e.g., for recording names and unique identifiers), integers (e.g., for recording sequences and the totals of countable entities *(parliamentary seats in the House of Commons)*, and real numbers (e.g., for recording numerical values to two places of decimal).

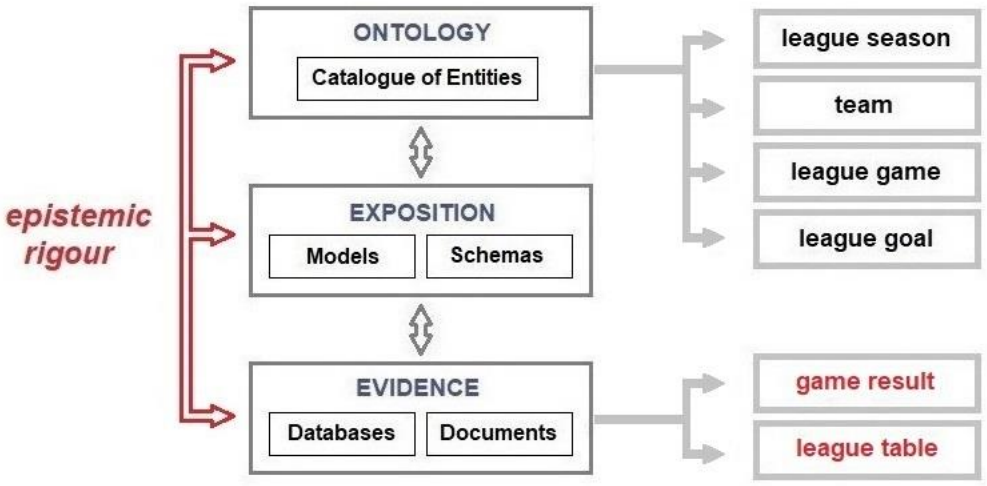
The list of names of football teams shown earlier is of entities that are usually known and referenced by their names. The trouble with names is that they can be changed, e.g., when there is a change of ownership of a company or team. This proclivity of humans to change names will be addressed at the right time because it is often necessary to reference information associated with an entity’s previous name. The ability to change an entity’s name does place an obligation on the custodians of certain types of databases to maintain records of previous names e.g., for a due diligence exercise, it may be useful to look up a company’s assets or liabilities when it was operating under a previous name.

An important feature of the databases discussed here is that the attributes of the entities may be designated as one of three types of ***keys***. This method of structuring and linking data is the invention of Ted Codd and Chris Date, pioneers of database technology in the 1960s. Keys are specifications used by some commonly available digital data management systems e.g., Microsoft Access, to facilitate the provision of structured information. Not to put too fine a point on it, these keys are the crux of the biscuit.

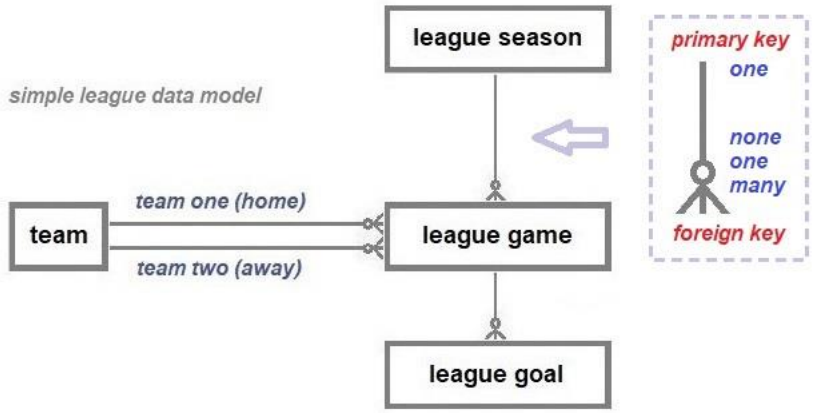
Keys are critical for the linking of the different tables that make up most databases and for the implementation of rules regarding the data being collected. It is horrifying to contemplate the chaos that will be unleashed if keys are not specified to set rules for a data collection exercise. However, keys are optional, and it is possible to structure and use data by other means and techniques e.g., customised documents such as journals and ledgers for financial transactions and, in the natural world, transfer RNA for genetic databases.

Chapter 2 – Designing a Database

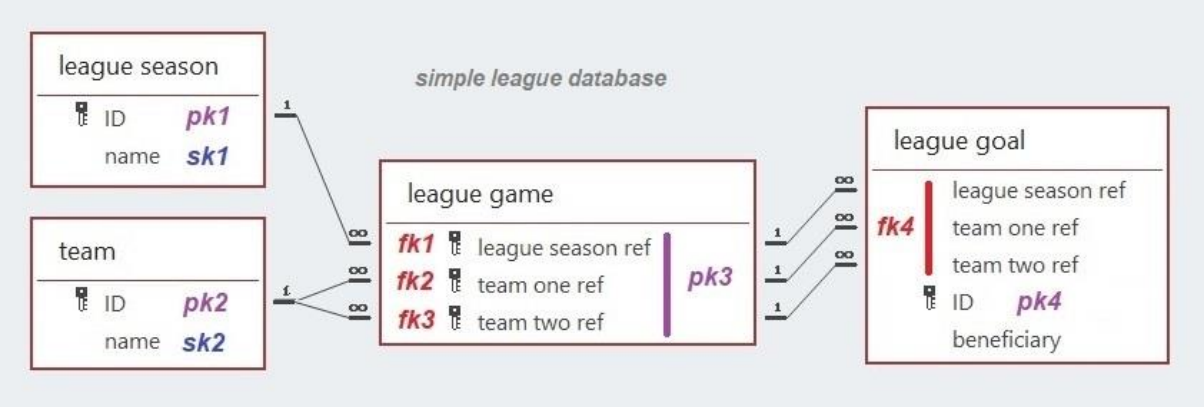
The following constructs will be used to specify the structure of a database; entities, attributes, data items, datatypes, and three types of keys. The three types of keys are ***primary***, ***secondary***, and ***foreign*** keys. The design process will outline how these constructs are deployed to build a data model *(the diagrammatic representation of the linked tables)*, and the utility of the resulting database will be assessed using sample data. This book will fail to meet one of its objectives if, at the end of the book, it is unclear why the databases discussed are structured as they are. This first example will discuss the design of a database to collect data about the games played and goals scored by teams competing over the course of a league season.



To illustrate the construction of a database, four data tables will be linked to enable information to be gleaned about the outcome of round robin football *(soccer – as in As**soc**iation Football)* games, a world-wide phenomenon and one that, I hope, is explicable to most adults *(young and old)*. In this example, teams play each other twice, at home *(team one)* and away *(team two)*. The record of the goals scored in each game is all that will be used to deduce the ranking of the teams at the conclusion of the season, e.g., the model does not allow for the removal of points from a team in the event of a breach of competition rules. The process of designing a database is iterative and may require revisiting and amending the data model as the testing of the integrity and usefulness of the database progresses.



The data model above is a minimalist model constructed to elucidate the nature of the links between the data tables and not to provide a model to satisfy the requirements of afficionados who wish to comprehensively document football league games e.g., the players, the substitutes, and the scorers. However, the model is extensible and can be extended to meet such additional requirements. Primary and foreign keys are the constructs necessary to link the tables.



A ***primary key (pk),*** whether it is a single attribute or a combination of attributes, must be unique, permanent, and unchangeable. The primary key must enable an entity to be uniquely identified when that key is referenced e.g., the combination of an individual's nationality and passport number can be used to identify that individual accurately because it will be unique. To establish a link, an entity records a reference *(the foreign key)* to another entity's primary key. There can only be a single primary key, i.e., only a single attribute or a single set of attributes can be designated as the primary key of an entity.

A ***secondary key (sk)*** is also a designated unique attribute or combination of attributes, but this key can be changed to a different unique value or combination of values. Unlike primary keys, an entity can have more than one secondary key. Secondary keys are used to implement rules, e.g., all teams in a football database must have a unique name as the users are more likely to use names to search the database as opposed to an assigned unique identifier, usually a meaningless character string.

A ***foreign key (fk)*** is the construct used to link the tables of data about the different entity types. An entity can have multiple foreign keys. The link between any two entity types is represented by a line

starting at entity type 1 and ending with a nought and a crow's foot *(or an infinity symbol)* at entity type 2. This type of link, a ***one-to-many***, specifies that there may be none, one or many type 2 entities associated with a single type 1 entity in a specific role and that the type 2 entities must be linked to one and only one type 1 entity,

The functions associated with the use of these keys are the only first principles needed to understand the rest of this explanation, especially the one-to-many relationship that is crucial to define the links between the tables and allow the aggregation of data items as required e.g., counting the total number of goals scored by a team over the course of a football game. The data model above is also specified as a machine-readable schema using a markup language, XML. That XML schema, discussed in chapter 5, can be safely ignored if the data model makes sense. The XML schema is additionally provided in the same helpful spirit that the Rosetta Stone was used to publish a decree about Ptolemy V's reign in three different scripts *(Egyptian hieroglyphs, Demotic and Ancient Greek)*. That thoughtful act helped both, Thomas Young and Jean-Francois Champollion decipher Egyptian hieroglyphs.

The provenance of data collected about sporting competitions can be verified by having been a spectator or by comparing media reports from multiple sources in the public domain. That is very unlike the process of verifying the personal details about individuals even if such data are in the public domain because, unless there is legal access to protected databases, it would be difficult to verify the accuracy of personal data.

The data used to test the *simple league database* in the next chapter are fictitious data. The use of confected data for testing purposes avoids the complications of using data collected by other individuals or commercial entities, data that may require payment for their use and/or be hard to verify. In this case it is the responsibility of the testing team to ensure that the test data covers all possible outcomes of football league games; home wins, away wins and draws including scoreless draws. Note that league football, unlike tournament football, allows draws, and therefore the *simple league* model does not cater for resolving draws e.g., by penalty shoot-outs.

A major responsibility for the custodians of databases is the rigorous testing of data processing software to ensure that it is fit for purpose, i.e., accurately updates a database. In 2022 the courts decided that the malfunctioning of the British Post Office's Horizon computer system between 2000 and 2014 led to a series of events that resulted in great harm to the lives of many sub-postmasters.

Entities must not be multiplied beyond necessity.

In keeping with the imperative expressed by ***Occam's razor***, a principle advocating parsimony, an attempt has been made to keep to a minimum the number of entity types required to meet the objectives of a data collection exercise. Furthermore, it is a truth universally acknowledged that simplicity and extensibility are a database designer's best friends *(see Gall's Law)*, and consequently simplicity and extensibility are presumed here to be useful design principles. The next stage is the

population of the *simple league database* with test data prior to generating the game results and league table.

Chapter 3 – A Simple Database

Shown below are two of the four tables and their attributes, with the primary and the secondary key annotated in each table. The league seasons and the teams are the named entities the data collection exercise is interested in. The other two entity types, the league games and goals are not referenced using names in this model. Returning to the problem with name changes, a meaningless string of characters has been added to the record of each name, a common construct to associate the name of an entity with a unique, permanent, and **unchangeable** attribute, commonly referred to as the unique identifier or ID*, (see below)*.

primary key 1 *secondary key 1*

ID	*name*
f5$m93	**CDL 2021**
h3%w4m	**EPL 2021/22**
vt1&8w	**CDL 2022**
n45LG!	**EPL 2022/23**

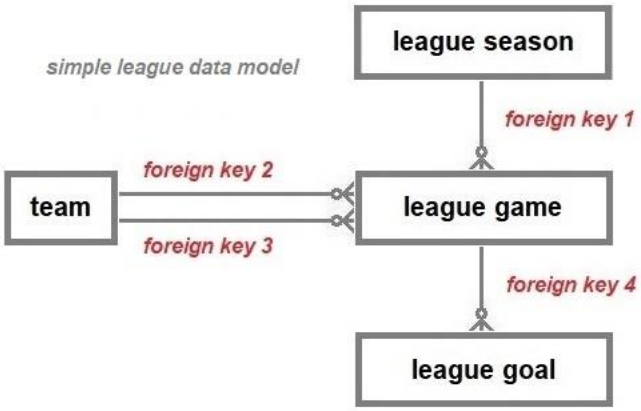
league season

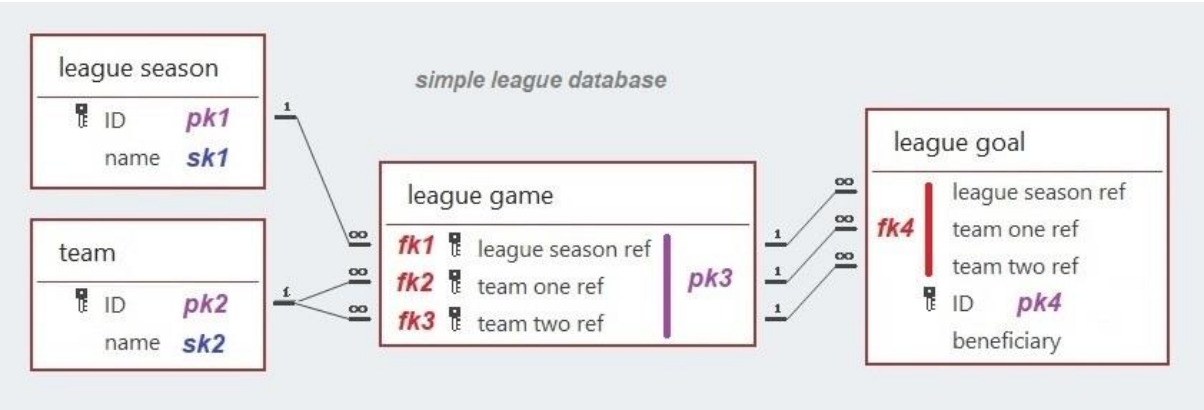
primary key 2 *secondary key 2*

ID	*name*
h6Ap%r	**falcons**
z1c(Y8	**eagles**
e%JiR4	**dolphins**
diT5£j	**sharks**

team

This attribute, the designated primary key, will be used to link to the other tables in the database as required and will allow the name of an entity to be changed while maintaining existing links to other entities. Other attributes about teams and league seasons that are of interest can be added if required. Keys may also be made up of multiple attributes. Furthermore, an attribute can be a part of more than one type of key simultaneously, as shown below.





The model shows an example of a multi-attribute primary key *(pk3)* comprised of three foreign keys *(fk1, fk2, and fk3)* for the *league game* table. That multi-attribute primary key is a multi-attribute foreign key *(fk4)* in the *league goal* table. The league season and the participating teams are linked by the *league game* entity. This is done by recording the primary key of the league season as an attribute of the league games, the *league season ref* attribute and the teams are similarly linked in the role of either the home team *(team one)* or the away team *(team two)*.

foreign key 1 *foreign key 2* *foreign key 3*

primary key 3

league season ref	*team one ref*	*team two ref*
f5$m93	h6Ap%r	z1c(Y8
f5$m93	h6Ap%r	e%JiR4
f5$m93	z1c(Y8	h6Ap%r
f5$m93	z1c(Y8	e%JiR4
f5$m93	e%JiR4	h6Ap%r
f5$m93	e%JiR4	z1c(Y8

league game

[IDs replaced by names]

league season	*team one*	*team two*
CDL 2021	falcons	eagles
CDL 2021	falcons	dolphins
CDL 2021	eagles	falcons
CDL 2021	eagles	dolphins
CDL 2021	dolphins	falcons
CDL 2021	dolphins	eagles

league game

The two tables above contain the same information *(the participating teams of the completed league games)* about the 2021 season of the Charles Darwin League (CDL). The first, the actual database table, displays the unique identifiers and the second is a report that has had the identifiers replaced by the current names of the season and the teams. The fictitious CDL 2021 is a league competition where three teams play each other twice, home, and away, over the course of a season.

The rule that teams in some leagues play each other just twice *(i.e., home, and away)* is used above to define the primary key of a league game. It is a rule that cannot be applied universally to all league seasons, e.g., to the 2022/23 and other recent Scottish Premiership seasons or the Isles of Scilly Football League, where teams play more than a single game against another team either at home and/or away. To accommodate such league competitions, league games will have to be differentiated using alternative or additional attributes.

[IDs replaced by names]

foreign key 4 *primary key 4*

league season	team one	team two	ID	beneficiary
CDL 2021	**dolphins**	**falcons**	**cz%g0F**	**team two**
CDL 2021	**eagles**	**dolphins**	**?e39vL**	**team two**
CDL 2021	**eagles**	**dolphins**	**4Tx2!M**	**team one**
CDL 2021	**eagles**	**dolphins**	**p40a@h**	**team one**
CDL 2021	**eagles**	**falcons**	**2q5H$w**	**team two**
CDL 2021	**eagles**	**falcons**	**L2d5&k**	**team two**
CDL 2021	**falcons**	**dolphins**	**@woL3v**	**team one**
CDL 2021	**falcons**	**eagles**	**b1G&3$**	**team one**
CDL 2021	**falcons**	**eagles**	**k7sl£2**	**team two**

league goal

Assume that there were nine goals scored over the course of the CDL 2021 season and that there was one game that ended goalless. Therefore, that game, the Dolphins *(home)* playing the Eagles *(away)* does not feature in the table of goals above. The only example of a non-key attribute shown is the attribute of greatest interest when reporting results, the *beneficiary* attribute *(i.e., who was the goal awarded to, team one or team two)*.

Each of the twelve types of data items that will populate the four database tables will be one of six distinct types of attributes, *ID*, *name*, *league season ref*, *team one ref*, *team two ref*, and *beneficiary*. The data items are all text strings.

To be of value to football fans, the tables above will need to be extended with data items recording the dates and venues of games, the team sheets for each game, the types of goals scored *(in-play, penalties, and own goals)*, the players who score and other statistics fans may wish to see. The *beneficiary* attribute will be redundant if information is recorded about goal scorers, the type of goal and the team sheets because the beneficiary can then be deduced from that additional information.

Before generating the results of the league games to verify the utility of the *simple league database*, it may be worth revisiting the words of George Box regarding models – “All models are wrong, but some are useful.” An uncompromising view in my opinion but a good reason to ensure that the specification of a model is rigorous as to what it represents, and why any known shortcomings should be clearly highlighted.

Chapter 4 – The Proof is in the Pudding

This example assumes that the league season, CDL 2021, has finished, and all six games have been completed. To produce mid-season league tables, it will be necessary to extend the model and the database to include data about the status of a league game, i.e., whether the game is yet to be played, now in play, or completed.

As you would expect, the *league goal* table is used to obtain the tallies for home and away teams *(by counting the IDs of the relevant goals in each game)*. Here, the two sets of tallies are called *game home tally* and *game away tally*.

game away tally

Field:	league season ref	team one ref	team two ref	tally two: ID	beneficiary
Table:	league goal	league goal	league goal	league goal	league goal
Total:	Group By	Group By	Group By	Count	Group By
Sort:					
Show:	☑	☑	☑	☑	☐
Criteria:					"team two"
or:					

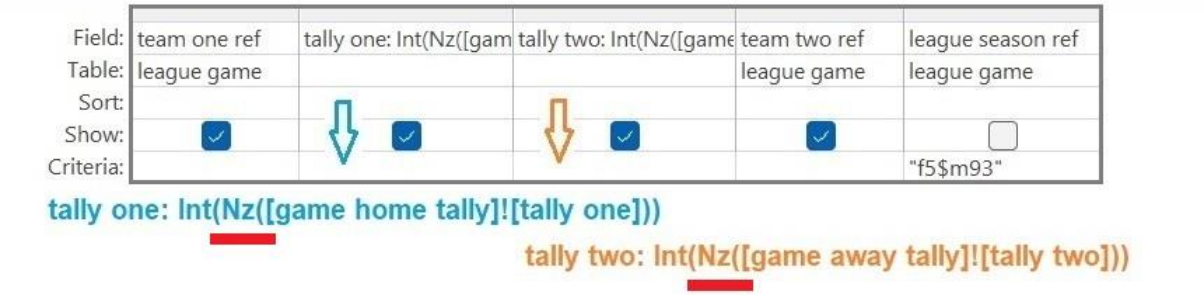
After the counts are completed, the four data items returned are the references to the league season, team one, and team two *(i.e., the three attributes of the primary key of a league game)* together with the total of the goals scored by either the home team or the away team, the total being dependent on the criteria specified for the count *(see above)*. The two tallies obviously make no mention of the teams that fail to score.

league season ref	*team one ref*	*team two ref*	*tally one*
f5$m93	**h6Ap%r**	**e%JiR4**	**1**
f5$m93	**h6Ap%r**	**z1c(Y8**	**1**
f5$m93	**z1c(Y8**	**e%JiR4**	**2**

game home tally

league season ref	*team one ref*	*team two ref*	*tally two*
f5$m93	**e%JiR4**	**h6Ap%r**	**1**
f5$m93	**h6Ap%r**	**z1c(Y8**	**1**
f5$m93	**z1c(Y8**	**e%JiR4**	**1**
f5$m93	**z1c(Y8**	**h6Ap%r**	**2**

game away tally


Field: team one ref | tally one: Int(Nz([gam | tally two: Int(Nz([game | team two ref | league season ref
Table: league game | | | league game | league game
Sort:
Show:
Criteria: "f5$m93"
tally one: Int(Nz([game home tally]![tally one]))
tally two: Int(Nz([game away tally]![tally two]))

To generate the results of games, a zero will need to be added as the tally of the non-scoring teams. Therefore, a function will need to be applied when combining the tallies so that a zero is returned when a team is goalless *(e.g., Microsoft Access's null zero function, Nz())*.

Charles Darwin League 2021

team one	*tally one*	*tally two*	*team two*
falcons	**1**	**1**	**eagles**
falcons	**1**	**0**	**dolphins**
eagles	**0**	**2**	**falcons**
eagles	**2**	**1**	**dolphins**
dolphins	**0**	**1**	**falcons**
dolphins	**0**	**0**	**eagles**

The English Premier League (EPL) with twenty teams has 380 games and probably over a thousand goals per season, hence why a fictitious league, with three teams, six games and nine goals, was used to illustrate the construction of a league database. The generation of league tables will not be discussed here as such calculations are well understood, at least since 1888 in England and probably earlier in Scotland.

The league table for the CDL 2021 season shown below is assembled by combining the home and away tables. Three points were awarded for a victory and a single point for a draw.

team one	pld1	win1	drw1	los1	pts1	GD1	for1	agn1
falcons	**2**	**1**	**1**	**0**	**4**	**1**	**2**	**1**
eagles	**2**	**1**	**0**	**1**	**3**	**-1**	**2**	**3**
dolphins	**2**	**0**	**1**	**1**	**1**	**-1**	**0**	**1**

team two	pld2	win2	drw2	los2	pts2	GD2	for2	agn2
falcons	**2**	**2**	**0**	**0**	**6**	**3**	**3**	**0**
eagles	**2**	**0**	**2**	**0**	**2**	**0**	**1**	**1**
dolphins	**2**	**0**	**0**	**2**	**0**	**-2**	**1**	**3**

CDL 2021 League Table

team	pld	win	drw	los	pts	GD	for	agn
falcons	**4**	**3**	**1**	**0**	**10**	**4**	**5**	**1**
eagles	**4**	**1**	**2**	**1**	**5**	**-1**	**3**	**4**
dolphins	**4**	**0**	**1**	**3**	**1**	**-3**	**1**	**4**

Chapter 5 – The Equivalent XML Schema

You can skip this chapter if the preceding four made sense. This chapter will provide the XML *(eXtensible Markup Language)* schema necessary to ensure that the league data in an XML document conform to the *simple league* model introduced earlier. A namespace *(an XML construct)* is required to designate those XML elements that represent the entities in the *simple league* model. The namespace used is ***sls*** *(simple league schema)*. For information about the use of XML, go to the ***World Wide Web Consortium***'s site at www.w3.org/standards/xml/, as there are appropriate tools and languages for providing data as web resources.

The data about the game between the Falcons and the Dolphins that were used in chapter 3 have been provided in an XML document. The document can have any number of the four different entity types as required. The outline of the XML document shows elements which are equivalent to the records from the database tables and have the same names as the tables.

```
<sls:ontology>
  <!--1 league season -->
  <leagueSeason>
     <ID>f5$m93</ID>
     <name>CDL 2021</name>
   </leagueSeason>
  <!--2 teams -->
  <team>
     <ID>h6Ap%r</ID>
     <name>falcons</name>
   </team>
  <team>
     <ID>e%JiR4</ID>
     <name>dolphins</name>
   </team>
  <!--1 league game -->
  <leagueGame>
     <leagueSeasonRef>f5$m93</leagueSeasonRef>
     <teamOneRef>h6Ap%r</teamOneRef>
     <teamTwoRef>e%JiR4</teamTwoRef>
   </leagueGame>
  <!--1 league goal -->
  <leagueGoal>
     <leagueSeasonRef>f5$m93</leagueSeasonRef>
     <teamOneRef>h6Ap%r</teamOneRef>
     <teamTwoRef>e%JiR4</teamTwoRef>
     <ID>@woL3v</ID>
     <beneficiary>team one</beneficiary>
   </leagueGoal>
 </sls:ontology>
```

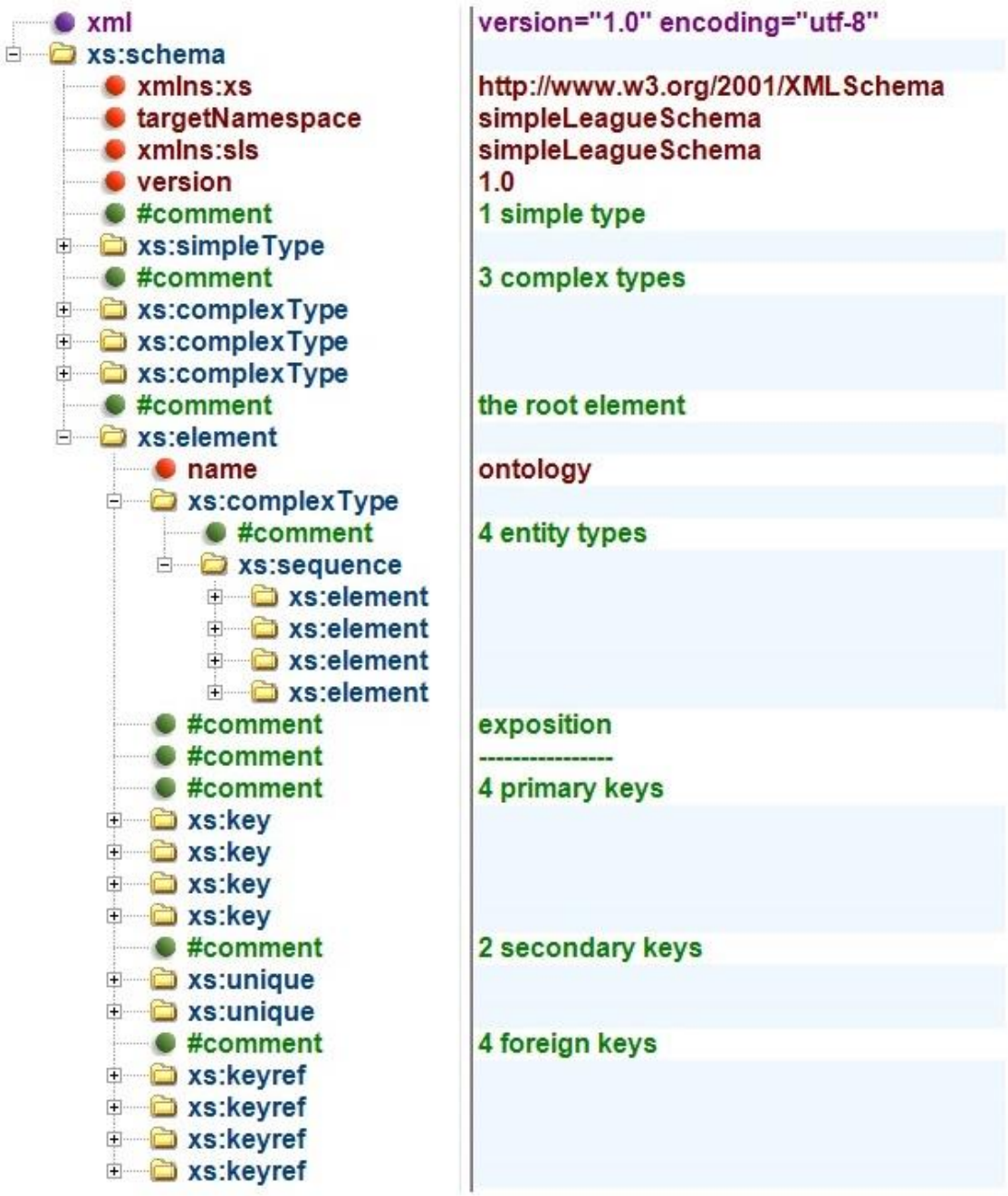
The above XML document was validated against the schema discussed below. The schema was validated using the CoreFiling XML Schema Validator. The schema that follows is equivalent to the model constructed in chapter 3.

The XML string datatype, ***xs:string***, is the only datatype used in the schema, i.e., the data items are all text strings. The string datatype is further constrained to obtain the specialised simple type that specifies the two options available when recording the *beneficiary* attribute of *league goals*.

There are three sections in this schema: simple types, complex types, and the root element. The simple types are used to construct other simple types and the complex types. The specialised simple type, *beneficiary option*, is used in the specification of the complex type, *league goal defn*. The complex types are used to construct other complex types and the root element. For the *simple league*

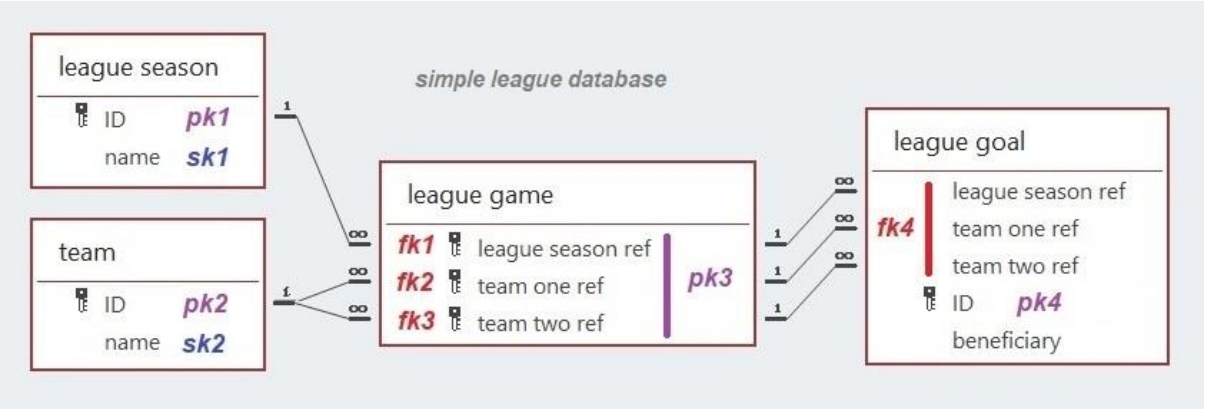
database, the root element contains the four entity types that will constitute the database. The complex type *named entity* is used to specify both, the *league season* and the *team* entity types in the root element. The complex type *league goal defn* is an extension of the complex type *league game defn.* The root element defines the elements that can be included in the XML document shown above.

The root element is made up of four elements that are listed together with their primary keys *(4) [aka key]*, secondary keys *(2) [aka unique]* and foreign keys *(4) [aka keyref]*. The four elements and the keys reflect the model described in chapter 3 i.e., the root element is the data model.



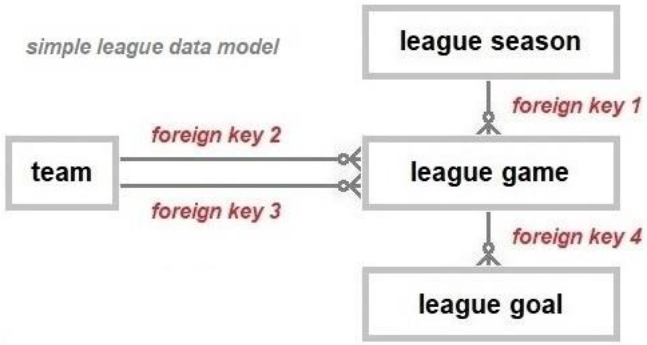
```
<xs:schema targetNamespace="simpleLeagueSchema" version="1.0">
  <!--1 simple type -->
  <xs:simpleType name="beneficiaryOption">
    <xs:restriction base="xs:string">
       <xs:enumeration value="team one"/>
       <xs:enumeration value="team two"/>
     </xs:restriction>
   </xs:simpleType>
  <!--3 complex types -->
  <xs:complexType name="namedEntity">
    <xs:sequence>
       <xs:element name="ID" type="xs:string"/>
       <xs:element name="name" type="xs:string"/>
     </xs:sequence>
   </xs:complexType>
  <xs:complexType name="lgeGameDefn">
    <xs:sequence>
       <xs:element name="leagueSeasonRef" type="xs:string"/>
       <xs:element name="teamOneRef" type="xs:string"/>
       <xs:element name="teamTwoRef" type="xs:string"/>
     </xs:sequence>
   </xs:complexType>
  <xs:complexType name="lgeGoalDefn">
    <xs:complexContent>
      <xs:extension base="sls:lgeGameDefn">
        <xs:sequence>
           <xs:element name="ID" type="xs:string"/>
           <xs:element name="beneficiary" type="sls:beneficiaryOption"/>
         </xs:sequence>
       </xs:extension>
     </xs:complexContent>
   </xs:complexType>

 <!--the root element -->
 <xs:element name="ontology">
   <xs:complexType>
     <!--4 entity types -->
     <xs:sequence>
        <xs:element name="leagueSeason" type="sls:namedEntity"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="team" type="sls:namedEntity"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="leagueGame" type="sls:lgeGameDefn"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="leagueGoal" type="sls:lgeGoalDefn"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
```

The root element is the place to specify the keys, the constructs necessary to enforce the rules governing the data and to link the different elements of the root element of the XML document.



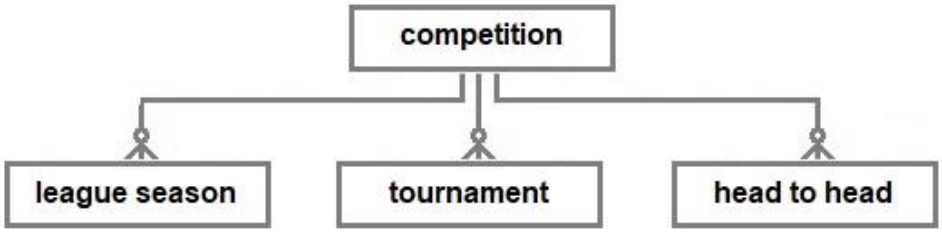

```
<!--exposition -->
<!----------------- -->
<!--4 primary keys -->
<xs:key name="primaryKey1">
   <xs:selector xpath="leagueSeason"/>
   <xs:field xpath="ID"/>
 </xs:key>
<xs:key name="primaryKey2">
   <xs:selector xpath="team"/>
   <xs:field xpath="ID"/>
 </xs:key>
<xs:key name="primaryKey3">
   <xs:selector xpath="leagueGame"/>
   <xs:field xpath="leagueSeasonRef"/>
   <xs:field xpath="teamOneRef"/>
   <xs:field xpath="teamTwoRef"/>
 </xs:key>
<xs:key name="primaryKey4">
   <xs:selector xpath="leagueGoal"/>
   <xs:field xpath="ID"/>
 </xs:key>


<!--2 secondary keys -->
<xs:unique name="secondaryKey1">
    <xs:selector xpath="leagueSeason"/>
    <xs:field xpath="name"/>
  </xs:unique>
<xs:unique name="secondaryKey2">
    <xs:selector xpath="team"/>
    <xs:field xpath="name"/>
  </xs:unique>
```

```
  <!--4 foreign keys -->
  <xs:keyref name="foreignKey1" refer="sls:primaryKey1">
     <xs:selector xpath="leagueGame"/>
     <xs:field xpath="leagueSeasonRef"/>
   </xs:keyref>
  <xs:keyref name="foreignKey2" refer="sls:primaryKey2">
     <xs:selector xpath="leagueGame"/>
     <xs:field xpath="teamOneRef"/>
   </xs:keyref>
  <xs:keyref name="foreignKey3" refer="sls:primaryKey2">
     <xs:selector xpath="leagueGame"/>
     <xs:field xpath="teamTwoRef"/>
   </xs:keyref>
  <xs:keyref name="foreignKey4" refer="sls:primaryKey3">
     <xs:selector xpath="leagueGoal"/>
     <xs:field xpath="leagueSeasonRef"/>
     <xs:field xpath="teamOneRef"/>
     <xs:field xpath="teamTwoRef"/>
   </xs:keyref>
 </xs:element>
</xs:schema>
```



Chapter 6 – Extending a Data Model

The *simple league* model can be extended to include two other types of competitions, multi-phase tournaments and head-to-head competitions *(i.e., between just two teams)*. Head-to-head competitions include Test series and competitions where the two participants qualify by winning other competitions e.g., the English Community Shield and the UEFA Super Cup. Furthermore, a competition may have both, a league season followed by a tournament to decide the prizes for the season. The English Football League Championship has had in recent years, a *(play-off)* tournament after the league season to decide the third team to be promoted to the Premier League. The top tier of professional Rugby League in England also employs play-offs.



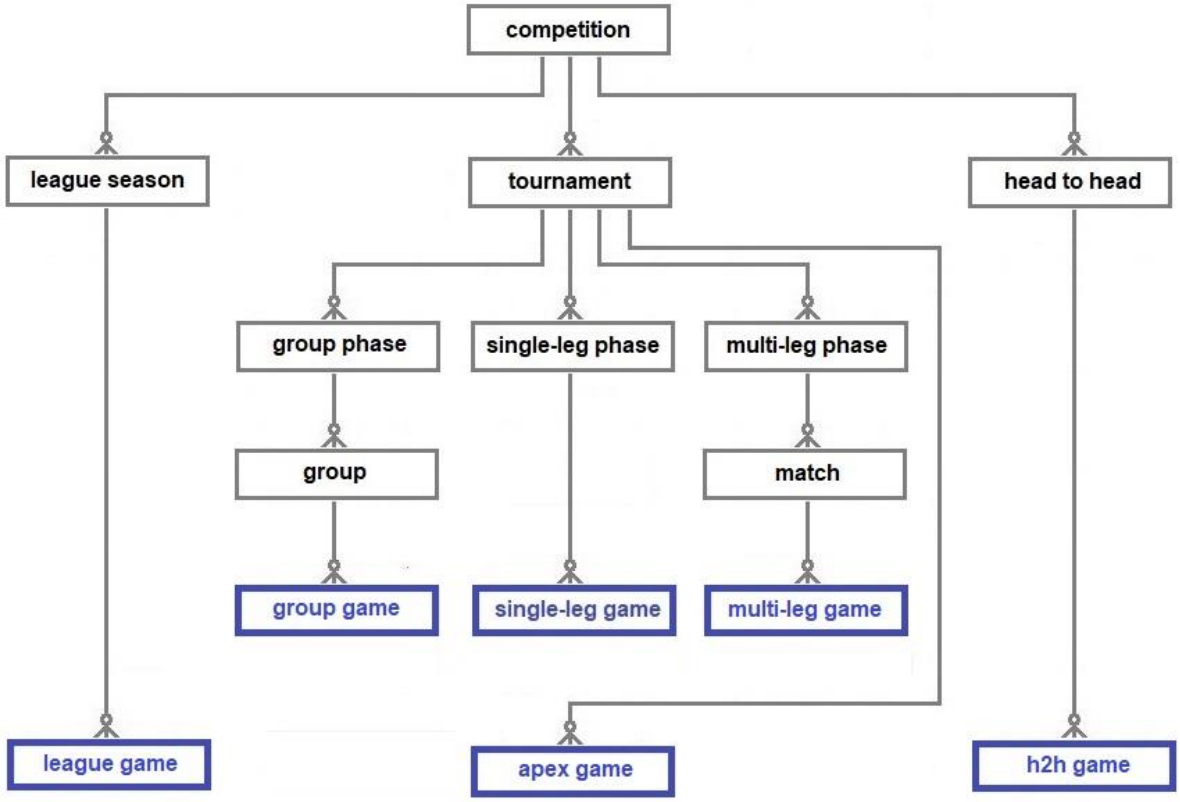
League seasons *(introduced by William McGregor of Aston Villa in 1888 to English football)* have a specific number of games related to them depending on the number of meetings between any pair of participating teams. Rugby union's Six Nations Championship has a single game between each pair, a total of fifteen games per season. Another round robin format, tournament groups, often have four teams playing a total of six games when the teams meet once *(FIFA World Cup)* and twelve games when they meet twice *(UEFA Champions League)*.

There are some competitions that apply rules that require amendments to the usual way outcomes are calculated. The Scottish Premier League *(SPL)* has employed an unorthodox mechanism for ensuring that twelve teams play thirty-eight games each during a league season. After three meetings between all the twelve participating teams, the SPL is treated as two separate leagues of six teams based on their league positions. There is then a single game between each of the six teams in each of the two leagues, a total of 228 league games for an SPL season. After the split, the SPL season becomes two separate leagues of six teams playing each other once, but with the teams retaining their points from before the split. The decision as to which of the teams play at home after the split is decided, presumably, from reviewing the three previous meetings between any two teams.

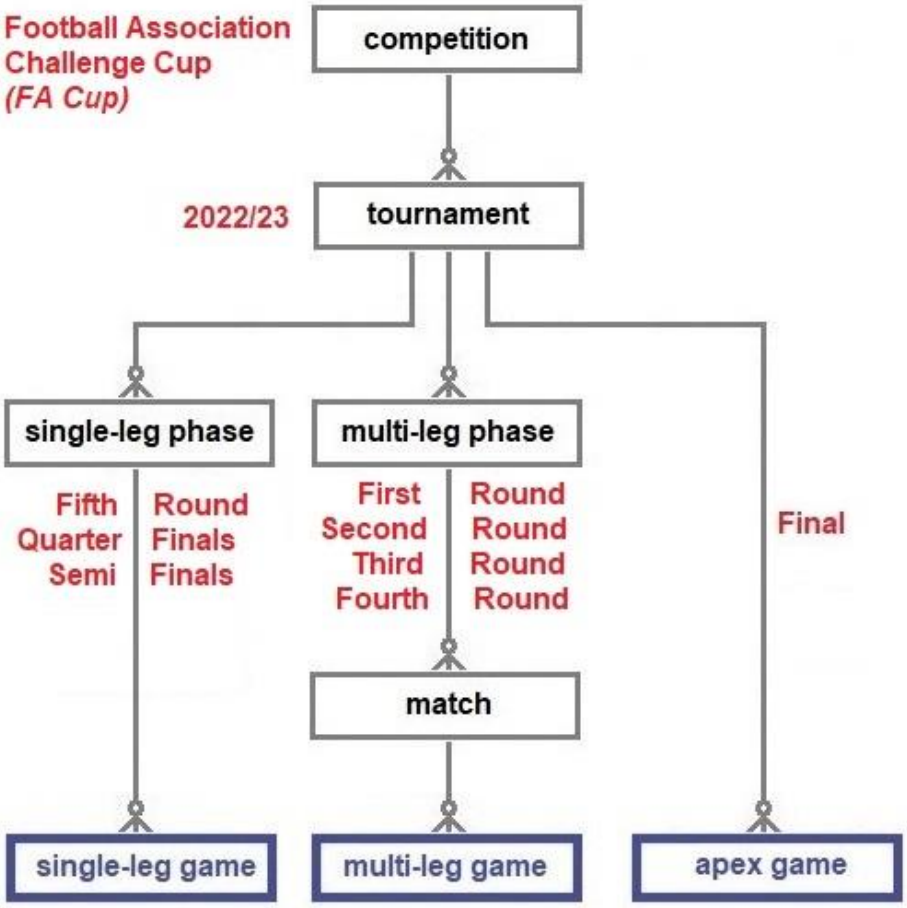
As intimated earlier, this means that the primary keys for the round robin games played in the SPL cannot be a combination of the identifiers of the league season and the two teams as used in the *simple league* model from chapter 3. *(For an even more convoluted round robin format, look at the 2023 Indian Premier League T20 cricket competition.))*

A tournament starts with four or more teams and then whittles the teams down with each successive phase, e.g., recent FIFA World Cup tournaments started with a group phase *(32 teams)*, followed by three single-leg knock-out phases *(16, 8 and 4 teams – octadic-finals, quarter-finals and semi-finals*

respectively) ending with the final and third-place games. The 2026 tournament has been expanded to include 48 teams. Tournament prizes are decided by the final and sometimes, the third-place game and/or the plate final, collectively known here as apex games. *(A plate tournament is a parallel tournament for teams eliminated at an early phase.)* Apex games are different from other types of games because there can only be one final, third-place decider, or plate final for a tournament.

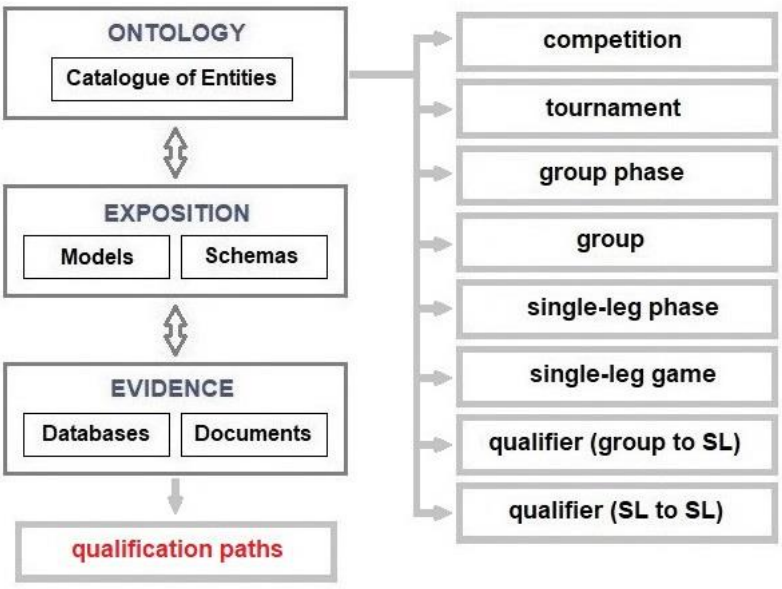


There is no option modelled here for tournaments to be decided by multi-leg apex matches as they are rare these days. As far as I know, the last time a multi-leg apex match decided a professional European football tournament was the 1997 UEFA Cup final between Schalke and Inter Milan over two games. The extended model is used below to describe the English 2022/23 FA Cup, the oldest football tournament *(from 1871/72)*, and the 2026 FIFA World Cup tournament.



The FA Cup tournament now begins with phases that allow re-plays in the event of draws and this model treats such matches as multi-leg matches. A drawn re-play is resolved by extra-time and, if necessary, a penalty shoot-out. The later phases, the Fifth Round onwards, are single-leg games that resolve draws immediately after the game with extra-time and shoot-outs.

The organisation of the 2026 FIFA World Cup tournament in North America will be analysed in the next chapter. The purpose of the exercise is to verify that the model above complies with the information provided by the organizers about the format of the tournament. The data published about a forthcoming tournament should enable a user to correctly plot the progress of the participating teams through the various phases. In this case it will require data about the qualification paths from the initial group phase via the single-leg phases to the apex games.



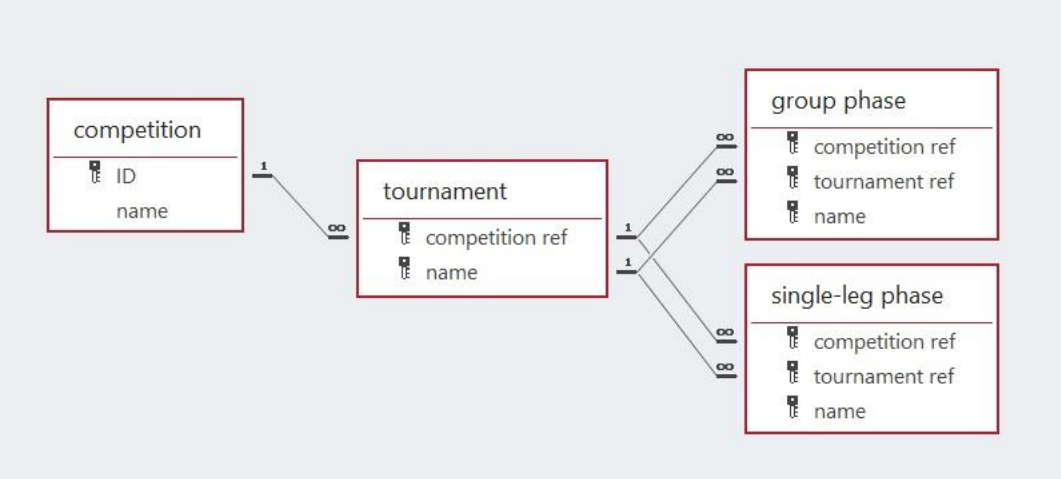
Carolus *(Father of Taxonomy)* Linnaeus's words below about the natural world provide an extremely useful insight about the analysis of databases.

The first step in wisdom is to know the things themselves; this notion consists in having a true idea of the objects; objects are distinguished and known by classifying them methodically and giving them appropriate names. Therefore, classification and name-giving will be the foundation of our science.

Systema Naturae (1735), trans. M. S. J. Engel-Ledeboer and H. Engel (1964)

Chapter 7 – FIFA World Cup 2026

While it may seem unnecessary to structure the data about a football tournament as a database when a single chart can do the same job competently, a database can be used to collect ***searchable*** data about multiple instances of a competition thus allowing reasonable comparisons e.g., of a team's performances over many different World Cup tournaments.



The 2026 tournament data used are that from the Bid United *(the successful North American bid)* site. For the 2026 tournament, to get to one of the two apex games *(final or third place game)*, a team will have to do well in the group phase and then win all its games until the semi-finals.

(N.B. the names used below for the tournaments and their components may not be what FIFA have used or may use but that is of no consequence in this analysis. However, the use of names instead of meaningless text strings as component-attributes of primary keys means that these names cannot be changed.)

ID	*name*
20fG£L	**Charles Darwin League**
a98#!D	**FA Challenge Cup**
d23%u=	**FIFA World Cup (men)**
n45Q+@	**English Premier League**

competition

competition ref	*name*
d23%u=	**2006**
d23%u=	**2014**
d23%u=	**2010**
d23%u=	**2026**

tournament

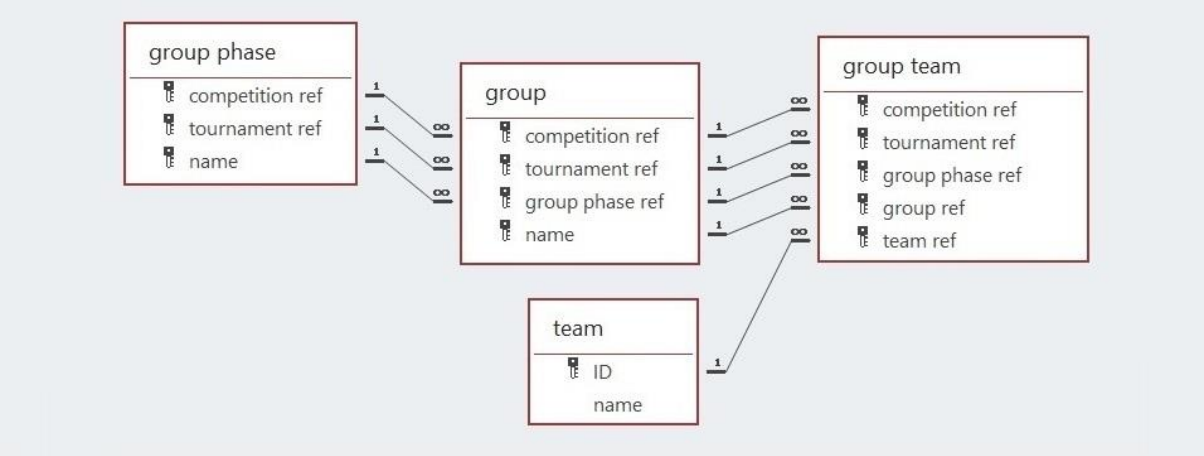
competition ref	*tournament ref*	*name*
d23%u=	**2014**	**Group Phase**
d23%u=	**2006**	**Group Phase**
d23%u=	**2026**	**Round 1**
d23%u=	**2010**	**Group Matches**

group phase

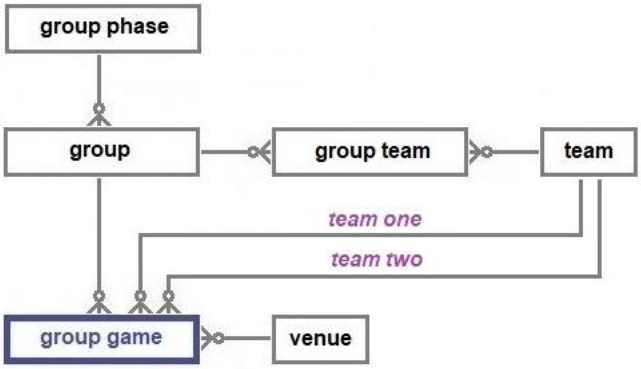
competition ref	*tournament ref*	*name*
d23%u=	**2006**	**Octadic finals**
d23%u=	**2026**	**Quarter-finals**
d23%u=	**2026**	**Round of 32**
d23%u=	**2026**	**Semi-finals**

single-leg phase

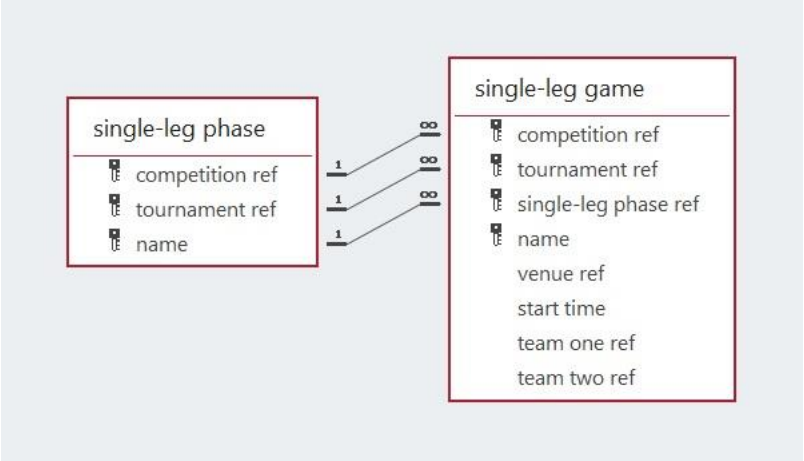
The 48 teams who will start the tournament will be assigned to 12 groups. Further information about the publication of the logistical details *(e.g., venues and start times of the 102 games and the allocation of the qualifiers to groups)* can be obtained from the Bid United site. To enable the specification of the criteria for progress, identifiers will have to be assigned to each group and single-leg game. This is because the two types of progress catered for by this model are that from a group to a single-leg game and that from one single-leg phase to the next.

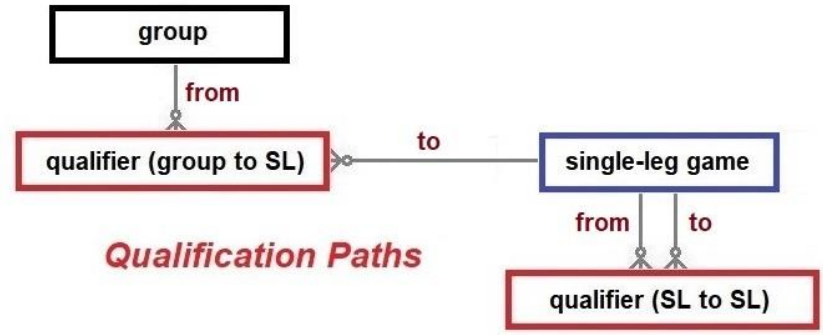


The twelve groups are assumed here to be named using the letters A through to L, a common naming convention. The model above represents the data structure necessary to record the teams in each group.

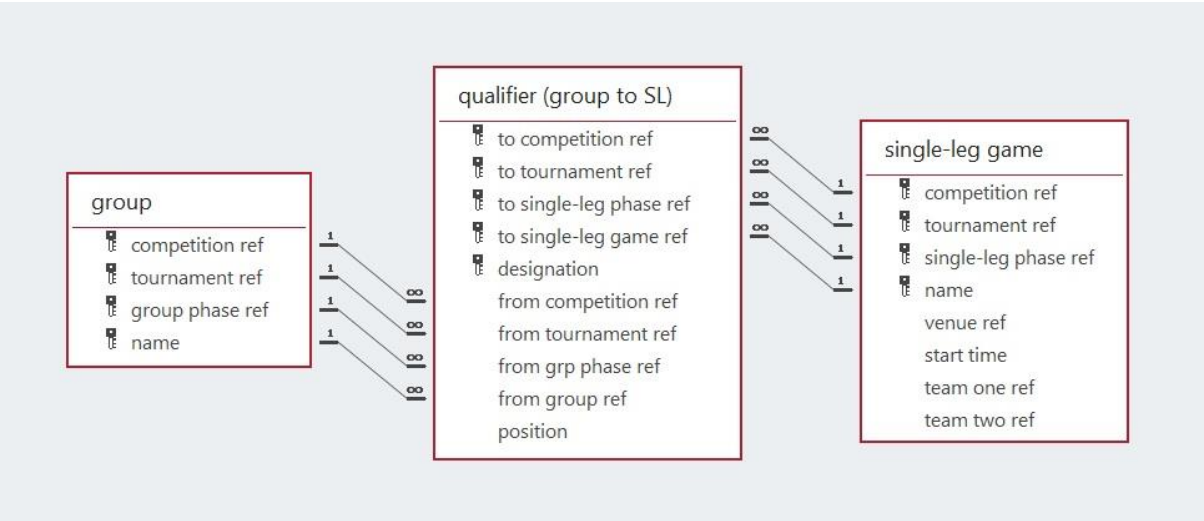


After the completion of the group phase, thirty-two teams progress to the first of the single-leg phases, the Round of 32. In this database, all games are named using the format “Game n” where n is an integer, e.g., Game 16.



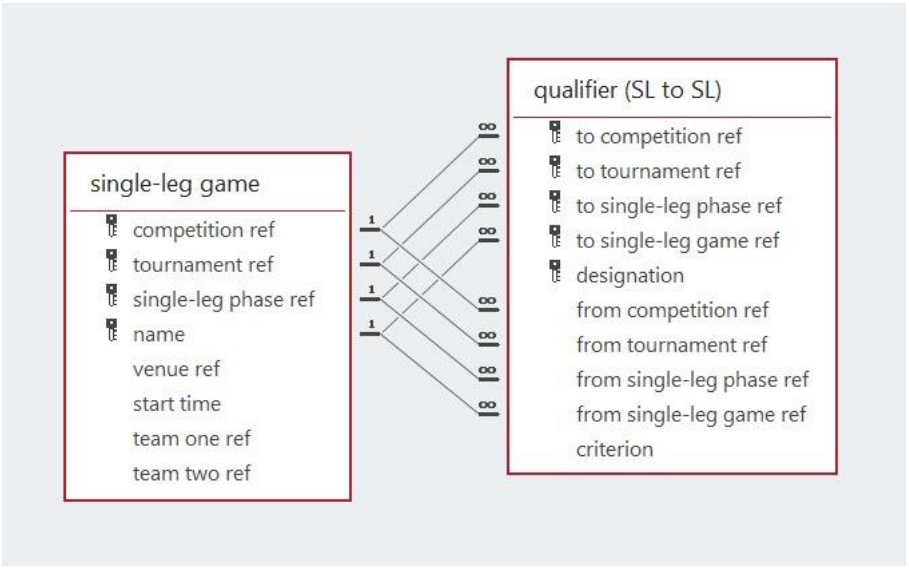


The model below represents the use of the group position *(winner, runner-up, or one of the best eight of the third-placed teams in the twelve groups)* necessary for a team to qualify to play in the single-leg phase of 32 knock-out games. The model also includes the specification of the qualifying team's designation as either *team one* or *team two,* even though the concept of home or away teams is not used in these tournaments. Eight of the thirty-two qualifiers for the Round of 32 can only be specified after the groups with the eight best performing third placed teams are known. The model and report that follows is that specifying the qualifiers from a single-leg game to the next single-leg phase. The reports below use test data as the required information has not been made available yet. The reports have also dispensed with the common attributes, i.e., the competition and tournament references.



to SL phase	to SL game	designation	from group	position#
Round of 32	**Game 16**	**team one**	**K**	**2**
Round of 32	**Game 16**	**team two**	**L**	**1**

2026 report: qualifier (group to single-leg game)

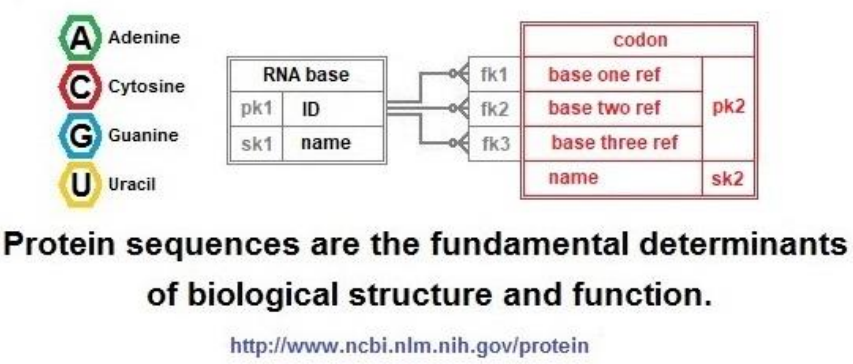


to SL phase	to SL game	designation	from SL phase	from SL game	criterion
Round of 16	**Game 8**	**team one**	**Round of 32**	**Game 15**	**winner**
Round of 16	**Game 8**	**team two**	**Round of 32**	**Game 16**	**winner**

2026 report: qualifier (single-leg game to single-leg game)

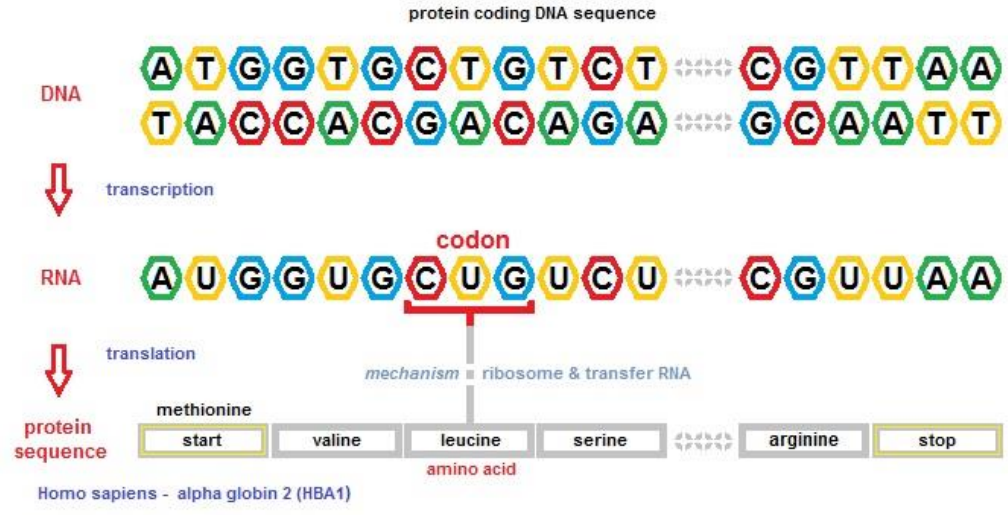
With respect to the apex games, while the self-evident qualification criteria from the semi-finals do not need to be specified, it will be necessary to designate the qualifying teams as either *team one* or *team two* for logistical reasons *(not shown)*.

Chapter 8 – Protein Sequences



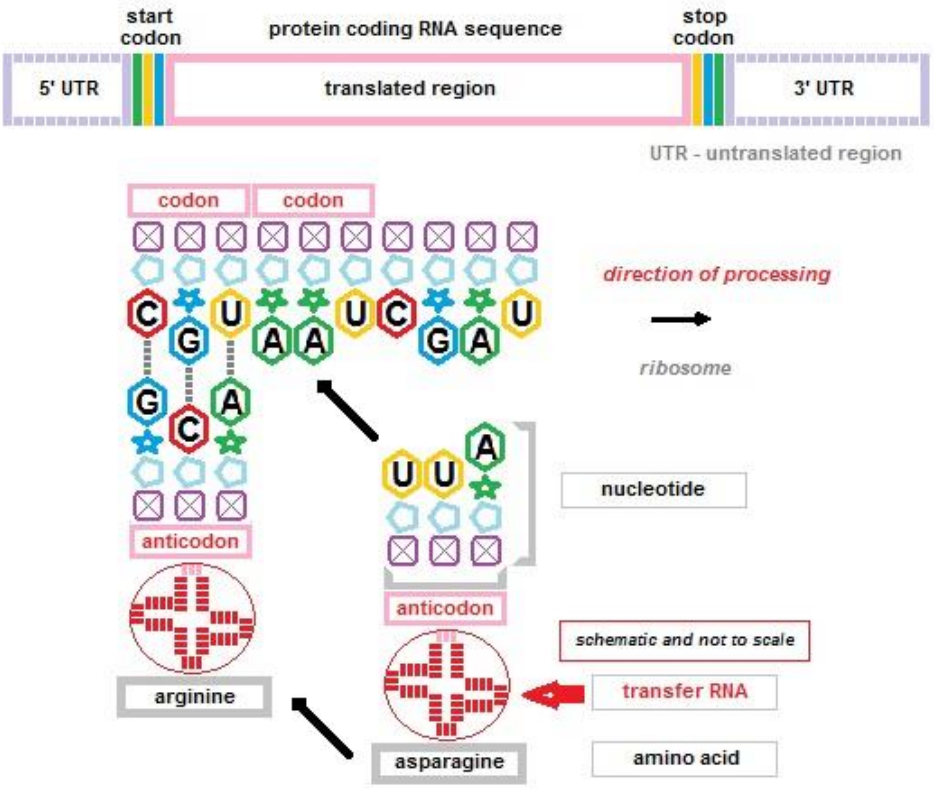
In 1902, Emil Fischer and Franz Hofmeister independently and simultaneously proposed that proteins are the result of the formation of peptide bonds between amino acids in a linear structure. In this discussion, the term ***protein sequence*** will be used to describe a linear unbranched chain of amino acids. This chapter provides information about the standard genetic code and the encoding of a human protein. It provides some of the evidence observed underpinning the 'central dogma of molecular biology' *(Francis Crick 1970)*, i.e., DNA makes RNA makes protein.

The model discussed here does not include the 24 known *(at the time of writing)* non-standard genetic codes nor the involvement of the exotic amino acids selenocysteine and pyrrolysine. It does, however, show how transfer RNA performs the function of a foreign key by implementing a relationship between a codon and an amino acid thus enabling the manufacture of protein sequences by ribosomes.



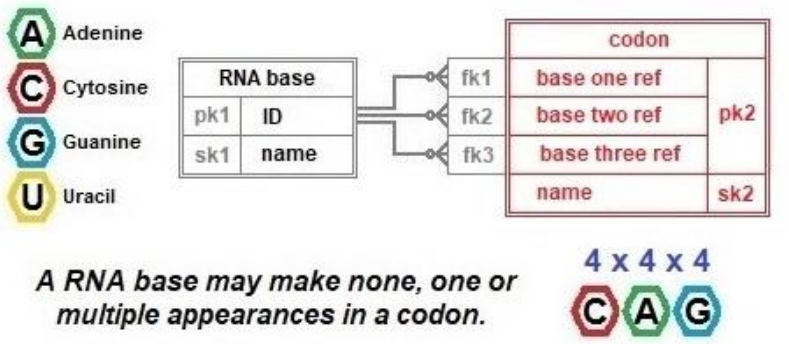
DNA encodes protein sequences and other genes e.g., functional RNA. The encoding of the sequence of amino acids in a protein sequence is done using the 4 DNA nucleobases, *(adenine, cytosine, guanine, and thymine)* arranged in groups of three, the codon. The codon defines the sixty-four arrangements that make the encoding mechanism explicable. Codons also include those that function as start and stop markers of the encoded protein sequences.

Marshall Warren Nirenberg, Har Gobind Khorana and Robert William Holley were awarded the Nobel Prize for Physiology or Medicine in 1968 for deciphering the genetic code. The encoded sequences of amino acids in DNA are transmitted from one generation of cells to the next via a number of different regenerative and reproductive techniques, all involving the replication of DNA, though with random mutations.



Transfer RNA associates each type of amino acid with an anticodon. The anticodon uses the complementarity of the bases *(A/U and C/G)* to add new amino acids to a growing chain of amino acids *(see above)* with the help of a ribosome until a stop codon is encountered on the messenger RNA.

The database proposed below has another illustration of the use of keys to enforce rules. This happens in the definition of a codon. To ensure that only sixty-four codons are specified in the database, a composite primary key using three foreign keys has been defined. The sixty-four possible codons may conduct a total of sixty-seven possible translations when the standard genetic code applies, albeit, depending on the context because of the dual functionality of three of the codons.



A codon may:

i **act as both, a start codon and an amino acid coding codon e.g. AUG, or**

ii **be recoded to perform a different function e.g. UGA, a stop codon, is recoded by the SECIS element to code for selenocysteine, or**

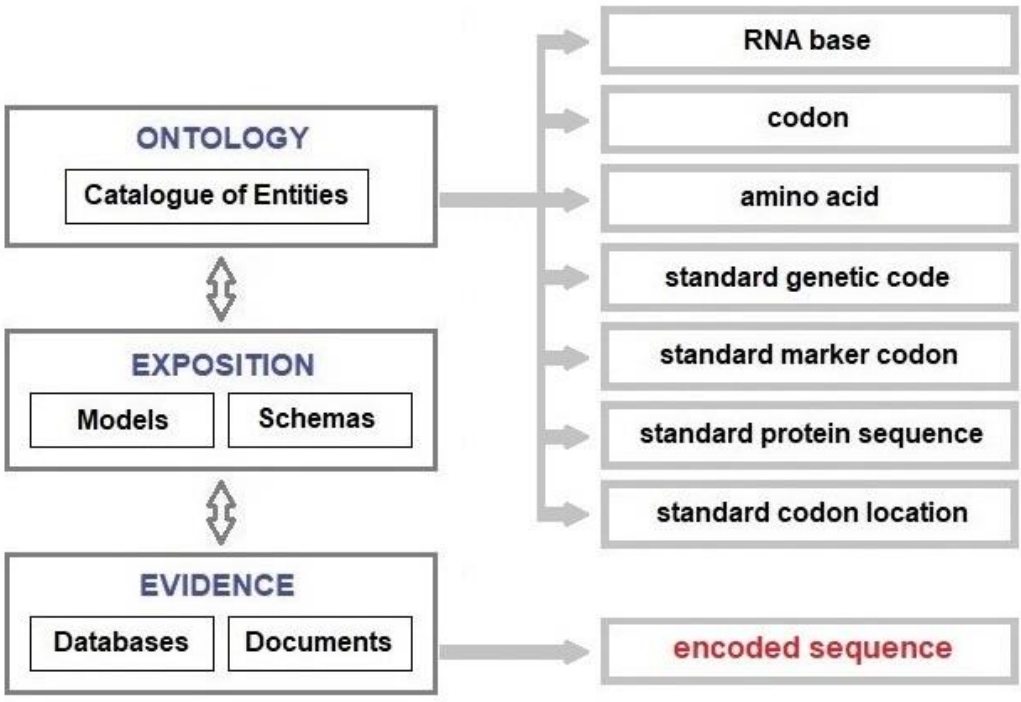
iii **perform a different function within a single organism or in different organisms as shown by the existence of multiple genetic codes e.g. the vertebrate mitochondrial code which is different from the standard code.**

As I have no idea of what is happening in academic or commercial genetic research, I have presumed that as far as non-specialists are concerned, the statements above are justified. *(The Guardian newspaper had a report of an artificial pair of protein bases 24 Jan 2017).* For information about genetic codes and protein encoding, I have relied pretty much exclusively on the **National Center for Biotechnology Information** ***(thank you NCBI, great material)***. I have assumed that the NCBI is a trusted authority on this subject.

As an autodidact on this subject, I may be exhibiting the Dunning-Kruger effect insofar as I may not be qualified *(i.e. without the required specialist knowledge)* to understand the consequences of not modelling other known, obscure and complex relationships between nucleotide sequences and the subsequent translation of codon sequences. That is a risk I took because the model proposed below is meant to explain the protein encoding mechanism and does not purport to describe all relationships between nucleotide sequences and protein synthesis.

On the other hand, you should, of course, independently verify all statements here *(nullius in verba)* before formulating any view as to the utility or otherwise of the model. Further investigation of non-standard codes and the recoding of some codons will require extensions or amendments to this model, one that is limited to the standard code only. However, even if my model is wrong or incomplete, it is reassuring to know that the mechanisms that transcribe DNA to RNA and then translate the messenger RNA into a sequence of amino acids are understood well enough by other individuals to enable some of them to design processes and tools to use DNA to store non-genetic data.

Chapter 9 – The Standard Genetic Code



Proteinogenic amino acids may be coded for by none, one or many codons. For example, serine is coded for by six different codons, tryptophan is only coded for by UGG and selenocysteine and pyrrolysine are coded for by none directly. Codons may also code either for start or stop markers.

The following two articles may throw more light on the matter of the non-standard behaviour of codons but there will be no further mention of the non-standard codes nor selenocysteine and pyrrolysine here.

Dual functions of codons in the genetic code *(Alexey V. Lobanov, Anton A. Turanov, Dolph L. Hatfield, and Vadim N. Gladyshev)*

Selenocysteine, Pyrrolysine, and the Unique Energy Metabolism of Methanogenic Archaea *(Michael Rother, and Joseph A. Krzycki)*

The objective of this data collection exercise is to report on the sequence of amino acids encoded in the DNA by translating the sequence of codons for a protein using the standard genetic code. The tables below provide the 4 RNA bases and some examples of amino acids and codons that feature in the explanation that follows.

RNA base

ID	name
c39!B9	uracil
p3q#H2	cytosine
wK@7:4	adenine
x$1K30	guanine

amino acid

ID	name
4p{E2?	phenylalanine
£Ybu9f	leucine
n6&o2H	isoleucine
D59s@y	methionine
6pa7%w	valine
bg3!>R	serine
nh$8y3	proline
6dK%&o	alanine
jT5n7<	tyrosine
3;g4-B	arginine

codon

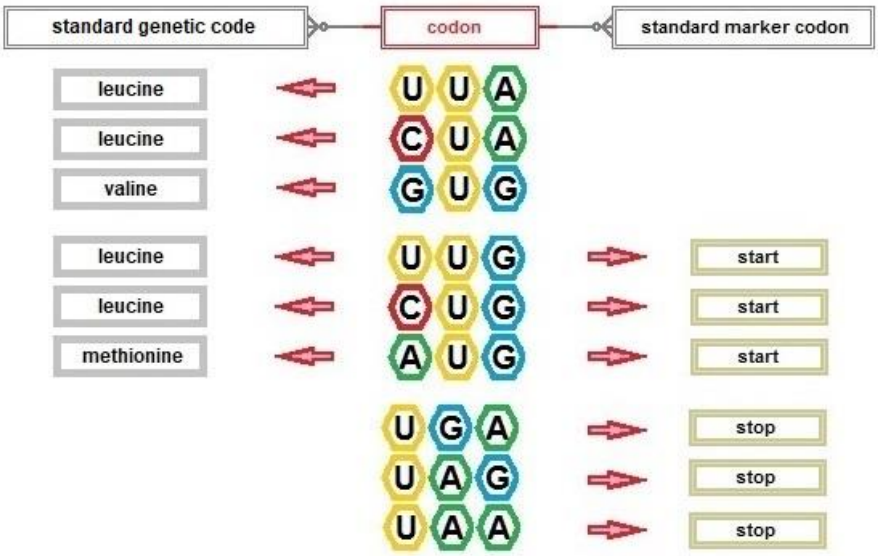
base one ref	*base two ref*	*base three ref*	*name*
c39!B9	c39!B9	c39!B9	UUU
c39!B9	c39!B9	p3q#H2	UUC
c39!B9	c39!B9	wK@7:4	UUA
c39!B9	c39!B9	x$1K30	UUG
c39!B9	p3q#H2	c39!B9	UCU
c39!B9	wK@7:4	p3q#H2	UAC
c39!B9	wK@7:4	wK@7:4	UAA
c39!B9	wK@7:4	x$1K30	UAG
c39!B9	x$1K30	wK@7:4	UGA
p3q#H2	c39!B9	wK@7:4	CUA
p3q#H2	c39!B9	x$1K30	CUG
p3q#H2	p3q#H2	c39!B9	CCU
p3q#H2	x$1K30	c39!B9	CGU
wK@7:4	c39!B9	x$1K30	AUG
x$1K30	c39!B9	x$1K30	GUG
x$1K30	p3q#H2	p3q#H2	GCC
x$1K30	x$1K30	c39!B9	GGU
x$1K30	x$1K30	p3q#H2	GGC
x$1K30	x$1K30	wK@7:4	GGA
x$1K30	x$1K30	x$1K30	GGG

phenylalanine	UUU	UUC				
leucine	UUA	UUG	CUU	CUC	CUA	CUG
isoleucine	AUU	AUC	AUA			
methionine	AUG					
valine	GUU	GUC	GUA	GUG		
serine	UCU	UCC	UCA	UCG	AGU	AGC
proline	CCU	CCC	CCA	CCG		
threonine	ACU	ACC	ACA	ACG		
alanine	GCU	GCC	GCA	GCG		
tyrosine	UAU	UAC				
histidine	CAU	CAC				
glutamine	CAA	CAG				
asparagine	AAU	AAC				
lysine	AAA	AAG				
aspartic acid	GAU	GAC				
glutamic acid	GAA	GAG				
cysteine	UGU	UGC				
tryptophan	UGG					
arginine	CGU	CGC	CGA	CGG	AGA	AGG
glycine	GGU	GGC	GGA	GGG		

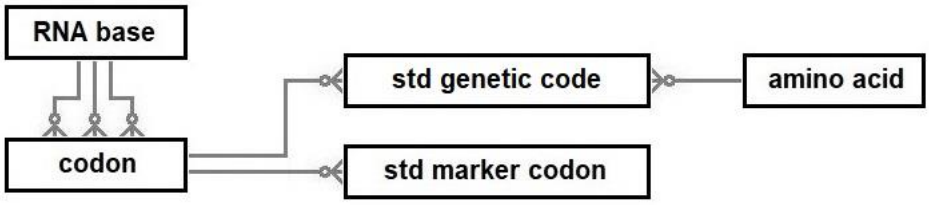
The Standard Genetic Code

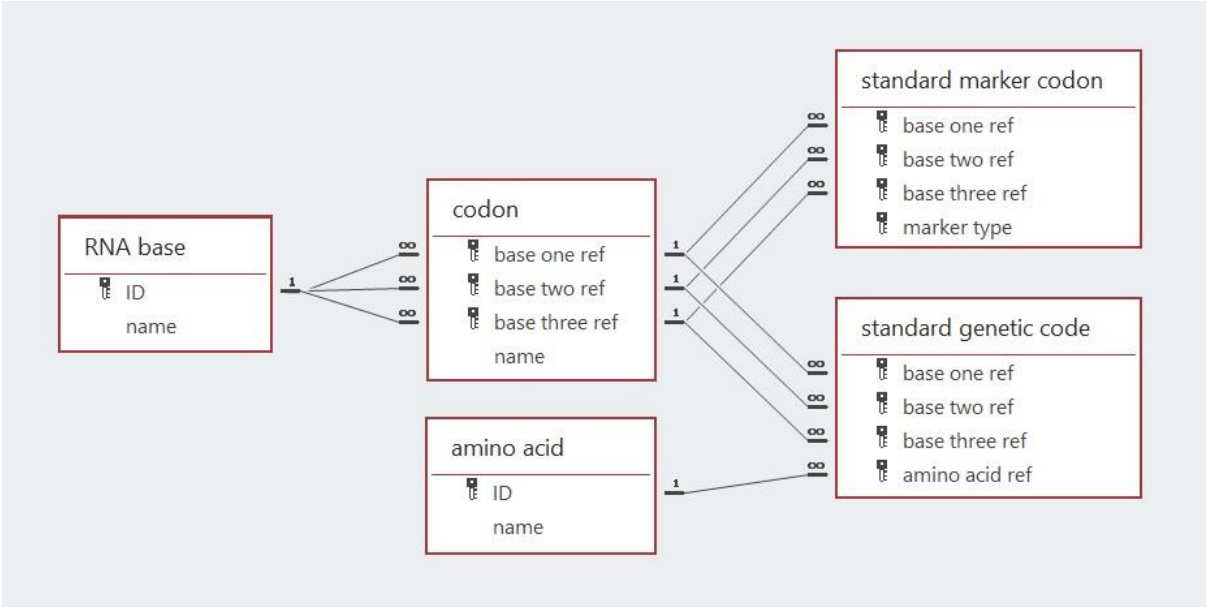
start marker	UUG	CUG	AUG
stop marker	UAA	UAG	UGA

The standard genetic code currently allows initiation from UUG and CUG in addition to AUG. Each of the three codons can function as both an amino acid code and as a start marker in the standard genetic code. This does not cause confusion during protein synthesis because after their first appearance in messenger RNA and before a stop codon is encountered, the three start codons will code for the amino acids shown below.



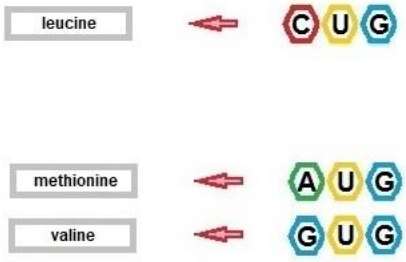
The above relationships will be implemented as two tables, the sixty-one ***standard genetic code*** table (*the transfer RNA role*) and the six ***standard marker codons*** table.





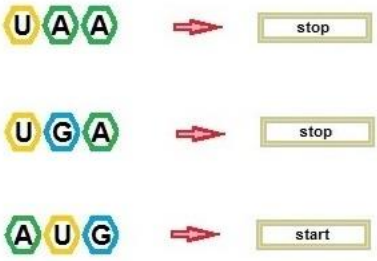
base one	*base two*	*base three*	*amino acid*
uracil	**cytosine**	**uracil**	**serine**
uracil	**adenine**	**cytosine**	**tyrosine**
cytosine	**uracil**	**guanine**	**leucine**
cytosine	**cytosine**	**uracil**	**proline**
cytosine	**guanine**	**uracil**	**arginine**
adenine	**uracil**	**guanine**	**methionine**
guanine	**uracil**	**guanine**	**valine**

standard genetic code



base one	*base two*	*base three*	*marker type*
uracil	**uracil**	**guanine**	**start**
uracil	**adenine**	**adenine**	**stop**
uracil	**adenine**	**guanine**	**stop**
uracil	**guanine**	**adenine**	**stop**
cytosine	**uracil**	**guanine**	**start**
adenine	**uracil**	**guanine**	**start**

standard marker codon



Chapter 10 – Encoded Protein Sequence

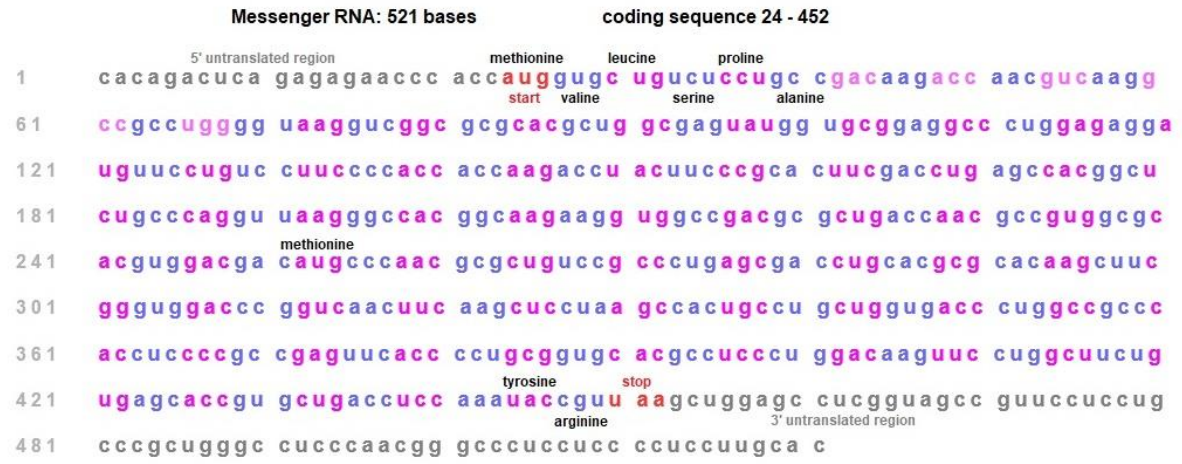
As discussed above, the sequence of amino acids in proteins can be deduced from the location of codons in protein coding nucleic acid sequences. I have categorised protein sequences into standard and non-standard protein sequences depending on whether the standard genetic code applies or if a non-standard genetic code applies. To illustrate how the sequence for a ***standard protein sequence*** is deciphered, I shall use sequence data from the NCBI for the messenger RNA of the human **HBA1** gene. The protein sequence is ***homo sapiens alpha-2 globin.***

source: *http://www.ncbi.nlm.nih.gov/nuccore/3859549*

ID	*name*
p!W45&	**homo sapiens alpha-2 globin (HBA-1)**

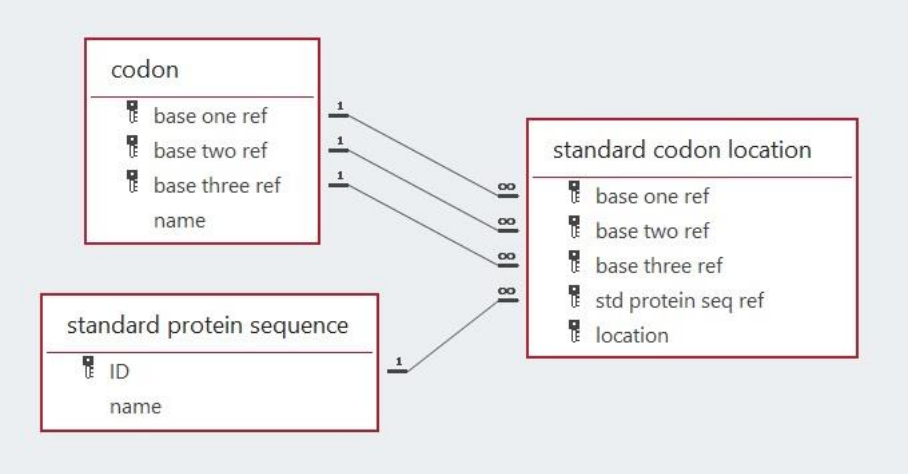
standard protein sequence

In the sequence below, I have greyed-out the initial and final strings of bases that comprise the 5' and 3' untranslated regions *(non-coding).* The sequence of 142 codons in the coding sequence starts with the AUG codon (*both a start codon and one that codes for methionine*), and the chain of amino acids terminates when the ribosome reaches the UAA codon. The translation of codons to amino acids for the sequence is also shown further below using a single letter to represent each of the twenty amino acids that may be coded for in the standard genetic code. The alternate convention is employed on the source page provided above.



```
  1   MVLSPADKTN
 11   VKAAWGKVGA
 21   HAGEYGAEAL
 31   ERMFLSFPTT
 41   KTYFPHFDLS
 51   HGSAQVKGHG
 61   KKVADALTNA
 71   VAHVDDMPNA
 81   LSALSDLHAH
 91   KLRVDPVNFK
101   LLSHCLLVTL
111   AAHLPAEFTP
121   AVHASLDKFL
131   ASVSTVLTSK
141   YR
```

alanine	A
cysteine	C
aspartic acid	D
glutamic acid	E
phenylalanine	F
glycine	G
histidine	H
isoleucine	I
lysine	K
leucine	L
methionine	M
asparagine	N
proline	P
glutamine	Q
arginine	R
serine	S
threonine	T
valine	V
tryptophan	W
tyrosine	Y



base one ref	*base two ref*	*base three ref*	*std protein seq ref*	*location*
c39!B9	**p3q#H2**	**c39!B9**	**p!W45&**	**4**
c39!B9	**wK@7:4**	**p3q#H2**	**p!W45&**	**141**
p3q#H2	**c39!B9**	**x$1K30**	**p!W45&**	**3**
p3q#H2	**p3q#H2**	**c39!B9**	**p!W45&**	**5**
p3q#H2	**x$1K30**	**c39!B9**	**p!W45&**	**142**
wK@7:4	**c39!B9**	**x$1K30**	**p!W45&**	**1**
wK@7:4	**c39!B9**	**x$1K30**	**p!W45&**	**77**
x$1K30	**c39!B9**	**x$1K30**	**p!W45&**	**2**

standard codon location

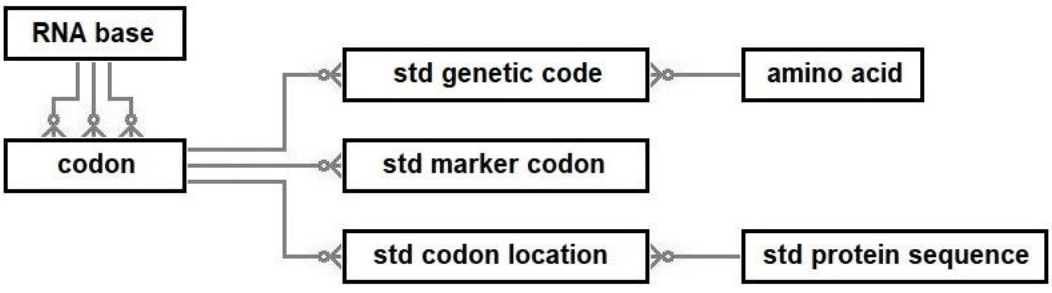
The table above records some of the locations of the codons for the protein product of the HBA1 gene. Although the amino acid reference is not present, the amino acid can be identified via the *codon* table followed by the *standard amino acid codon* table. The table below is a report of the codon sequence with the corresponding amino acid sequence also provided, a more human-friendly version.

standard protein sequence	*location*	*amino acid*	*codon*
homo sapiens alpha-2 globin (HBA-1)	**1**	**methionine**	**AUG**
	2	**valine**	**GUG**
	3	**leucine**	**CUG**
	4	**serine**	**UCU**
	5	**proline**	**CCU**
	77	**methionine**	**AUG**
	141	**tyrosine**	**UAC**
	142	**arginine**	**CGU**

report: encoded sequence

That covers the remit of this model of the encoding of standard protein sequences in nucleic acids. The transcription and translation of nucleic acids to protein sequences are, in my opinion, staggeringly wondrous phenomena and that they now seem somewhat explicable to humans is a cause for celebration with the finest wines available to humanity.

It should be borne in mind that the data model of the encoding of protein sequences discussed here has not been reviewed by anyone other than the author.

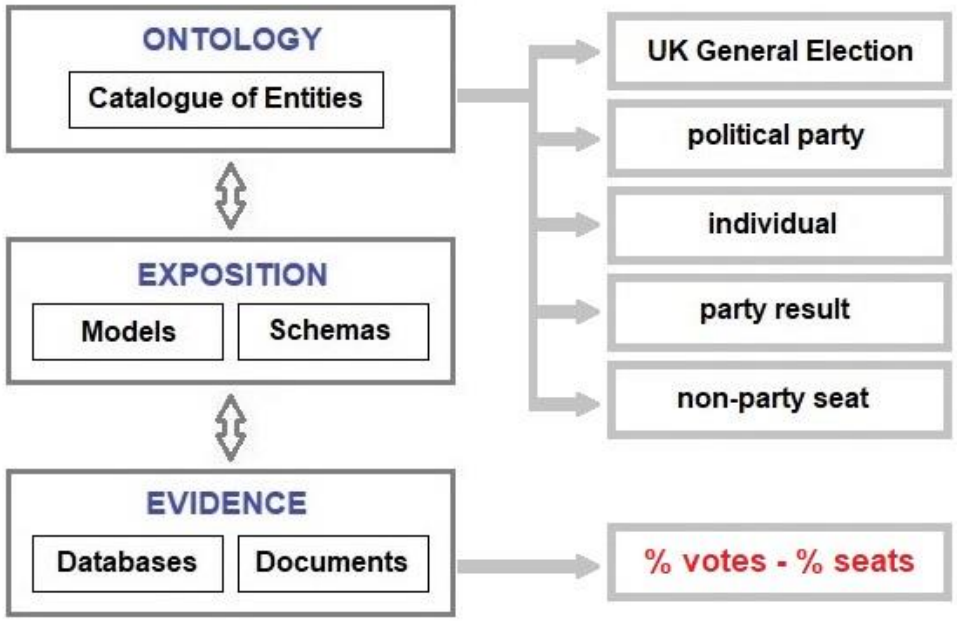


Chapter 11 – UK 2019 Electoral Data

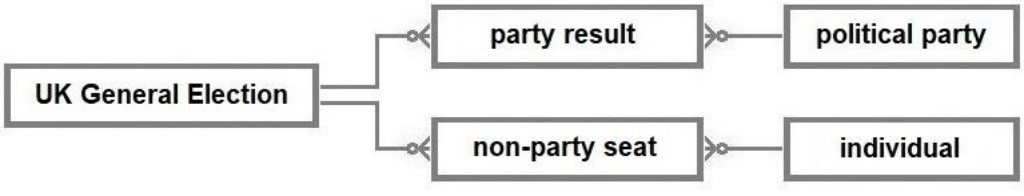
The databases discussed up to this point were populated with raw data, e.g., recording the goals scored and the sequence of protein bases in nucleic acids. This chapter will use derived data to summarise the UK's December 2019 General Election *(GE)* results, i.e., the results of 650 constituencies to see if it is an accurate reflection of the electorate's votes for the UK's political parties. When using derived data, an ability to verify their accuracy is a critical consideration. This discussion presumes that the House of Commons *(HoC)* Library is a reliable source of derived data about the results of UK parliamentary elections. What does engender confidence in the data is the fact the HoC Library also provides the raw data that were aggregated to obtain the derived data used in the election report. The data used are from this House of Commons Library Briefing Paper.

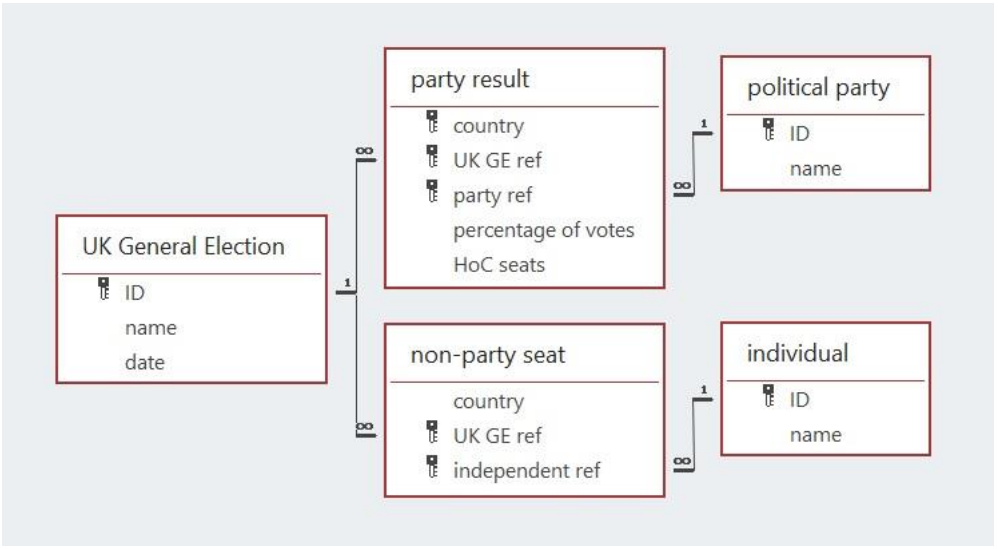
The four countries of the United Kingdom *(England, Scotland, Wales, and Northern Ireland)* hold general elections periodically to simultaneously elect Members of Parliament *(MP)* to the HoC in 650 single-member constituencies, The voting system used is called First Past the Post *(FPTP)* and requires the electorate to choose a single candidate from those standing. The winner is the candidate who obtains a plurality of the valid votes cast in a constituency. Some political parties in the UK field candidates in just one country e.g., the Scottish National Party and Sinn Fein, and some in more than one country e.g., the Conservatives and Labour.

At the 2019 GE, England had 533 seats, Scotland 59, Wales 40, and Northern Ireland 18. Of the 3320 candidates in the 2019 GE, the vast majority were affiliated to political parties. None of the 224 independent candidates won a seat. The MP who is the Speaker of the HoC is a special case and is listed separately, regardless of any previous party affiliation. Some parties do not stand a candidate in the Speaker's constituency at a general election. A candidate can only stand in a single constituency at a UK Parliamentary general election. Of course, things can change after a general election and there will be MPs who may have resigned a party whip, had the whip withdrawn, or defected to another party, in addition to new members joining because of by-elections or an MP taking the Chiltern Hundreds. The current state of the parties can be found at this HoC site, https://members.parliament.uk/parties/Commons.



For this exercise, the items of interest are the percentages of the votes each political party amassed *(the party result)* in each of the four countries and the number of seats won in the House of Commons. For comprehensiveness, all independent, i.e., unaffiliated MPs will also be recorded, in this case, only the Speaker. The efficacy of a parliamentary electoral system is assumed here to be correlated to its ability to ensure that there is minimal discrepancy between the proportion of votes garnered by a political entity and the proportion of its representation in parliament.





There are electoral systems, other than FPTP, collectively termed proportional representation *(PR)* e.g., party lists or using transferable votes with ranked choices in multi-member constituencies, which have been adopted in many countries to reduce such discrepancy. Unfortunately, even if a PR electoral system is used as opposed to FPTP, there is academic work *(Arrow's Impossibility Theorem)* showing that it is a Sisyphean task to attempt to get to a perfect correlation between the votes cast and the representation. This theorem however did not discourage many countries from adopting systems that attempt some form of proportional representation to enhance equal representation for votes.

Speaker - Lindsay Hoyle	
Conservative	**365**
Labour	**202**
Scottish National Party	**48**
Liberal Democrat	**11**
Democratic Unionist Party	**8**
Sinn Fein	**7**
Plaid Cymru	**4**
Social Democratic and Labour Party	**2**
Green Parties	**1**
Alliance	**1**

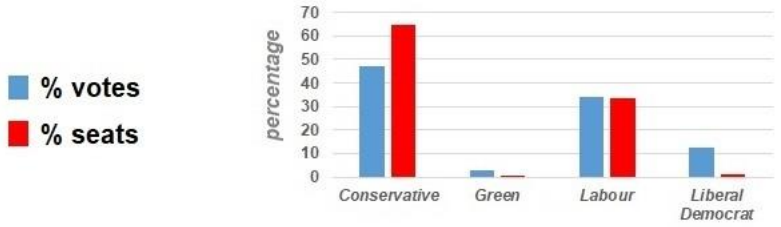
UK General Election 2019 Results

Although the UK parliamentary elections are often reported by country, only the overall result matters. The table above is an aggregated version of the results of the four separate countries of the UK with the Speaker's seat excluded. The Conservatives, with a clear majority, went on to form the government. The tables below have disaggregated the information above and assigned the percentage of seats and of votes by country. They show that the parties with the largest share of the seats in each country have achieved that with a percentage of the votes that is lower than the percentage of seats. The Speaker's constituency, Chorley, is an English constituency and that is why the total of English seats do not sum to a 100%.

All the valid votes have not been accounted for in the following statistics because the votes for parties or individuals who did not gain a seat have been excluded. The HoC Library Briefing Paper and the associated detailed constituency data mentioned earlier do provide information about the unsuccessful parties and independents.

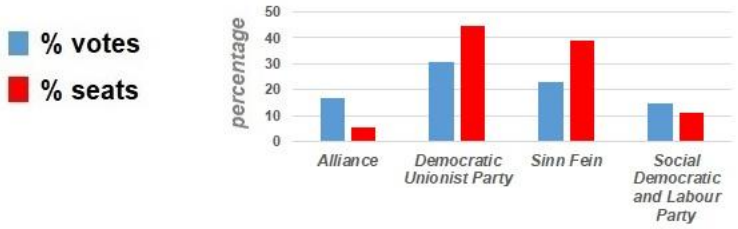
England: UK GE 2019

	% votes	% seats	seats
Conservative	47.2	64.7	345
Green Parties	3.0	0.2	1
Labour	33.9	33.6	179
Liberal Democrat	12.4	1.3	7
total	*96.5*	*99.8*	*532*



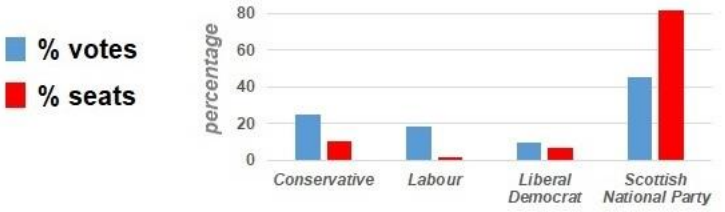
Northern Ireland: UK GE 2019

	% votes	% seats	seats
Alliance	16.8	5.6	1
Democratic Unionist Party	30.6	44.4	8
Sinn Fein	22.8	38.9	7
Social Democratic and Labour Party	14.9	11.1	2
total	*85.1*	*100.0*	*18*

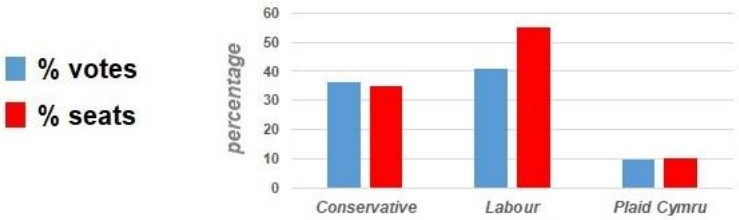


Scotland: UK GE 2019

	% votes	% seats	seats
Conservative	25.1	10.2	6
Labour	18.6	1.7	1
Liberal Democrat	9.5	6.8	4
Scottish National Party	45.0	81.4	48
total	*98.2*	*100.0*	*59*



Wales: UK GE 2019	% votes	% seats	seats
Conservative	36.1	35.0	14
Labour	40.9	55.0	22
Plaid Cymru	9.9	10.0	4
total	*86.9*	*100.0*	*40*



Before coming to any conclusions about UK politics based on the information above, you should make sure that you are satisfied with the provenance of the data.

Chapter 12 – Open Contracting Data Standard

The purpose of this chapter is to discuss the use of an international data standard. The government of the United Kingdom is committed to publishing public procurement contracts using the **Open Contracting Data Standard** (OCDS) to, inter alia, enable members of the public to search for, download, and analyse data about ***public procurement transactions***. The government's public procurement policy is published online. The policy materials include the legal framework that buttresses the policy *(e.g. the Public Contracts Regulations 2015 (PCR 2015))*, handbooks and other guidance, amendments or changes to the policy.

In the interests of transparency, the Open Contracting Partnership (OCP) says, "… a publication policy is an essential data guide for users. Each OCDS publisher ought to have a publication policy ,,,":

- **What information is and is not covered within the scope of the publication**
- **Any custom codelists used - or custom codes added to existing codelists**
- **How timely or frequent the publication is**
- **Where the data comes from and how it is generated**
- **The formats of the publication and where to access the data**
- **Any differences in the contracting processes or fields covered by different formats or access methods**
- **License information for data reuse**
- **Contact information of the publisher**

According to the OCDS, there are broadly four mutually exclusive transaction types available to public sector buyers, the choice being dependent on the relevant legislation and the circumstances of each transaction. There are many factors that can introduce complexity to a procurement transaction. The OCDS lists them as *hard cases*.

Hard cases	
Unsuccessful processes	Buyers and suppliers
Framework agreements	Organization references
Pre-qualification and pre-selection	Organization identifiers
Awards and contracts	Personal identifiers
Electronic catalogs	Organizational units
Updates and amendments	Organization classifications
Milestones	Beneficial ownership information

A valid financial transaction typically must satisfy the following criteria, authenticity, integrity, non-repudiation, consistency, and availability.

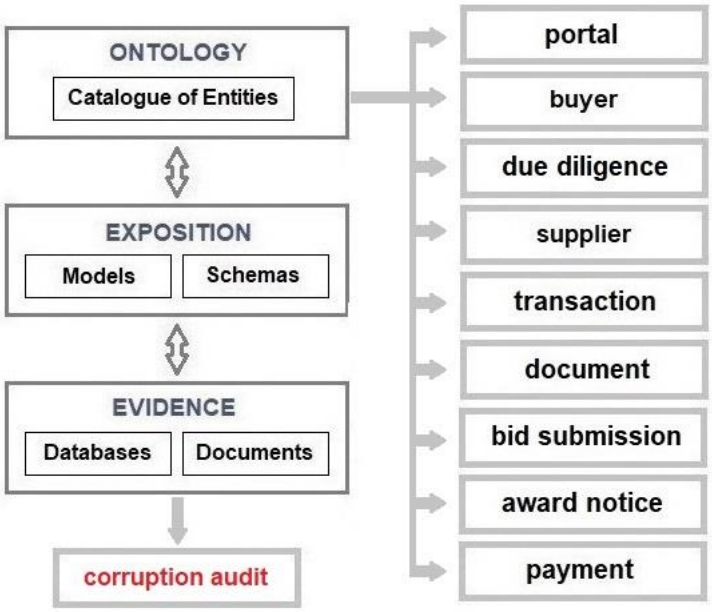
Authenticity	**The transaction must be initiated by an authorised party or parties with the necessary permissions or access rights to carry out the transaction.**
Integrity	**The transaction data must be accurate and complete, and it must not be tampered with or altered in any way during the transaction process.**
Non-repudiation	**The transaction must be verifiable and traceable, so that it cannot be denied by any party involved in the transaction.**
Consistency	**The transaction must comply with all applicable laws, regulations, and standards, and it must be consistent with the terms and conditions of any relevant contracts or agreements.**
Availability	**The transaction data must be available and accessible to authorised parties in a timely and secure manner.**

The fact that contract data are available to the public raises the issue of confidentiality related to some details of the participants' commercial activities. A balance must be found between what is required to be published in accordance with the OCDS transparency imperatives and the need for confidentiality. In 2018 the Information Commissioner's Office published a report assessing the transparency gap in public procurement.

There are separate public contracting portals for England, Scotland, Wales, and Northern Ireland. An OCDS compliant public procurement portal must have an identifier assigned by the OCP called the ***ocid***. The ocid for England's electronic government procurement portal, ***Contracts Finder***, is **ocds-b5fd17**.

This narrative will focus on the processes and documents associated with a transaction that allow members of the public to satisfy themselves as to the governance of that transaction, e.g., all transactions must have the known or declared conflicts of interests documented and published. The OCDS lists "beneficial ownership information" as a possible cause of a *hard case*.

Buyers publish their requirements on a public sector procurement portal, e.g., *Public Contracts Scotland*, and suppliers then bid for contracts to meet those requirements in a tender process. The next stage is the receipt and evaluation of bids followed by the award of contracts to the successful bidder(s). It is likely that most high-value public procurement transactions will have been preceded after due diligence was conducted on prospective suppliers and those findings documented for the benefit of auditors. In exceptional circumstances a buyer may award a contract without competition. The denouement is the delivery of the procured items and payments in accordance with the provisions of the contract(s).



When a tender is not open to all suppliers, e.g., the procurement of some types of military equipment or personal protective equipment (PPE) during a public health emergency, there should be a justification posted by the buyer explaining why all or some competition has been dispensed with. The data made available should be sufficient to enable a member of the public to exercise their judgement as to the integrity and efficacy of a transaction, e.g., whether the lack of tendering is reasonable or should be further investigated.

The four OCDS method codes below define the four methods that can be used to conduct a public procurement transaction. The OCDS allows for extensions and amendments to these codes to meet local requirements but does impose rules on how changes are applied. In the case of the method code below, all extensions can only be subsets of the four types shown. The OCDS terms this a ***closed codelist***. In England's *Contracts Finder*, the direct method below is categorised as "other."

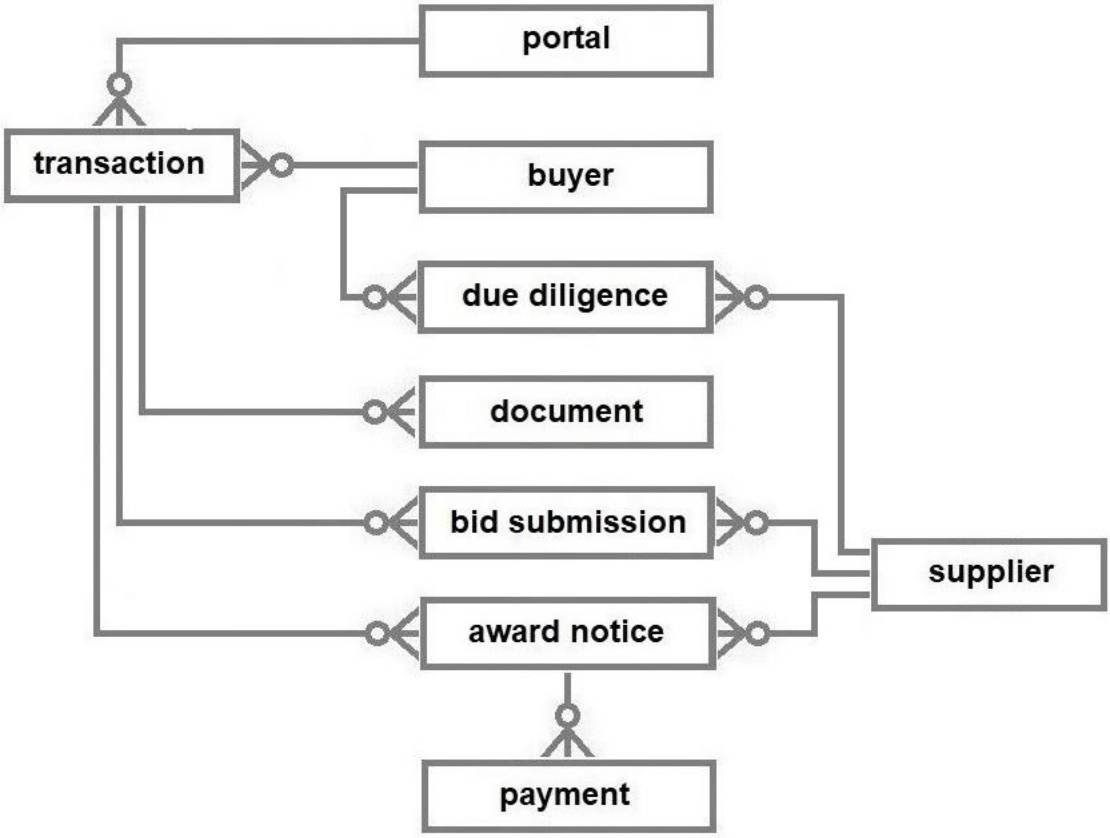
OCDS Code	*Description*
open	**All interested suppliers may submit a tender.**
selective	**Only qualified suppliers are invited to submit a tender.**
limited	**The procuring entity contacts a number of suppliers of its choice.**
direct	**The contract is awarded to a single supplier without competition.**

There are many functions supported by the OCDS that will not be discussed here unless they impinge directly on the governance of a transaction. There are many codelists that are critical for the logistics of managing a transaction *(e.g., milestone type)* and to understand industrial trends, *(e.g., the Common Procurement Vocabulary (CPV) for product classification)* but they will not feature in this discussion. The OCDS document type codelist is an open codelist and the administrators of the portal are responsible for ensuring that the buyers have the option of posting the different types of documents that facilitate and evidence the procurement process locally.

Buyers, upon initiating a transaction, must publish the documents in accordance with the timescales stipulated in the relevant regulations or guidelines. The combination of the method code of a transaction and the document type codes will allow an auditor to check if the appropriate documentation has been published, e.g., a direct award should have a justification document published in addition to the specification of the deliverables and payment schedules as agreed between the buyer and the supplier in the contract. It would be reasonable to expect the publication of the outcome of the due diligence on a supplier, especially with a direct award, in the interests of transparency and the avoidance of the stench of corruption.

document type
due diligence
qualification criteria
invited suppliers list
justification
conflict of interest
tender
bid
award
contract
invoice
receipt

The document types listed above are critical to the process of providing the requisite evidence for auditors. There are, of course, many more document types involved in the management of the processes supported by the OCDS, but they are not required for this exercise. While the majority of document types above are common to all tendered procurement processes, the *invited suppliers list, due diligence documentation, qualification criteria,* the *justification*, and the *conflicts of interest* are requirements for public procurement transparency.



It is assumed in this explanation that the public are primarily interested in ***what*** was procured from the public purse and ***how much*** was paid for it. Whether an auditor is able to ascertain if a transaction was a fair transaction and that the public had obtained "value-for-money" will depend on the information published about the transaction on the government's electronic procurement portal, industry knowledge and any further research the auditor undertakes.

The initiation of a transaction is assumed here to be when a tender notice or, exceptionally, an award notice is published. However, in some jurisdictions, there are regulations that require other events during a public procurement transaction to be documented e.g., in Paraguay, the planning decisions have to be evidenced.

The auditor will need access to any changes made to any published document e.g., to the contracts especially if there are additional charges or if there have been delivery problems. All OCDS compliant solutions must have facilities to publish subsequent amendments to any published data.

This model is my idea of a data model representing the minimum types of data that are sufficient to effectively reach a conclusion as to the integrity of a public procurement transaction, i.e., achieve OCDS compliance. The model does not purport to be representative of any real-world solution. It only discusses those facets of the OCDS that are related to audit facilities and ignores the logistical part of the public procurement process.

The data available on *Contracts Finder* about the direct award by the Department of Health and Social Care of a contract to Ayanda Capital Limited in 2020 for the supply of PPE will be reviewed next. The reason that this transaction is being analysed is because the government's transparency

policy, *Contracts Finder*, and *Ayanda* all featured in a high court judgement *(41-page pdf – e.g., see para. 2, 4, and 31(a))* in February 2021, (Neutral Citation Number: [2021] EWHC 346 (Admin)).

Although no individual procurement decision was challenged in the claim brought by the Good Law Project and three opposition MPs, D. Abrahams, C. Lucas, and L. Moran against the Secretary of State for Health and Social Care, the claim was about the government's implementation of its public procurement transparency policies and principles and matters that arose as a result of the public sector procurement activities. In my opinion, this judgement is a useful exegesis of current public procurement policy documentation in the UK.

To find the award notice on *Contracts Finder*, the following search terms were used. The keyword used was "Ayanda" and the procurement stage "awarded contract" was selected. The only result of the search is shown below.

https://www.contractsfinder.service.gov.uk/Search/Results

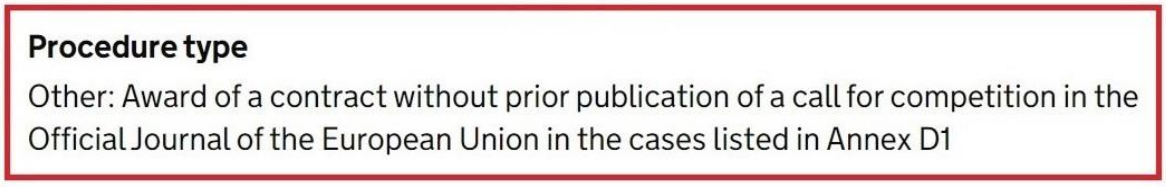
Supply of Personal Protective Equipment for Healthcare W...

DEPARTMENT OF HEALTH AND SOCIAL CARE

The Department of Health and Social Care (the "Authority") has awarded a contract to **AYANDA** CAPITAL LTD for the supply of supply of Personal Protective Equipment (masks) to protect Health and Care workers from contracting the Covid 19 disease.

Procurement stage Awarded contract
Contract location United Kingdom
Awarded value £252,500,000
Awarded supplier **AYANDA** CAPITAL LTD
Publication date 27 July 2020, last edited 4 September 2020

The link to the award notice is embedded in the title of the result. The publication date of the contract award notice implies that the transaction had been initiated sometime before 27 July 2020, as will be confirmed from the information in the Contract Award Notice *(CAN - as the award notice is termed in the judgement of Chamberlain J referenced earlier)*. This CAN was last amended on 4 September 2020 but there are no data as to what was edited. The CAN specifies that the method or procedure type is the equivalent of the OCDS *direct* method.


Procedure type

Other: Award of a contract without prior publication of a call for competition in the Official Journal of the European Union in the cases listed in Annex D1

The CAN provides links to other resources including two documents, the justification for the lack of competition and the contract. There were no documents related to due diligence or conflicts of interest published on *Contracts Finder* with respect to this transaction. A link to the specification of the

deliverables, embedded in the contract document, was not working *("not found" was the response)* on 10 August 2022 and 17 August 2022. There were no documents posted about payment information related to this transaction. Therefore, excluding the missing specifications, the data available in the public domain about this transaction are in three documents, the award notice or CAN, the justification of the absence of competition, and the contract.

The justification for this transaction is a short document *(less than 500 words)* and it references PCR 2015 Regulation 32(2)c which authorises a no competition award under certain circumstances, The auditor has to decide whether the justification is adequate. The contract is a document *(17-page pdf)* of three parts: the contract details, the terms, and conditions, and two schedules with some redactions. Again, the auditor has to decide if the information available in the redacted contract was sufficient to arrive at an informed conclusion about the governance of the transaction.

On 1 September 2022, another supplier who was awarded two direct contracts for PPE by the DHSC was in the news. The search results are shown below. There were no documents posted about either any conflicts of interest or about any payments related to these two transactions - PPE Medpro CAN 1/ PPE Medpro CAN 2.

Supply of Personal Protective Equipment for Healthcare W...

The Department of Health and Social Care

The Department of Health and Social Care (the "Authority") has awarded a contract to PPE **Medpro** for the supply of Personal Protective Equipment (masks) to protect Health and Care workers for contracting the Covid 19 disease.

Procurement stage Awarded contract
Contract location SW1H 0EU
Awarded value £80,850,000
Awarded supplier PPE **Medpro** Limited
Publication date 20 October 2020

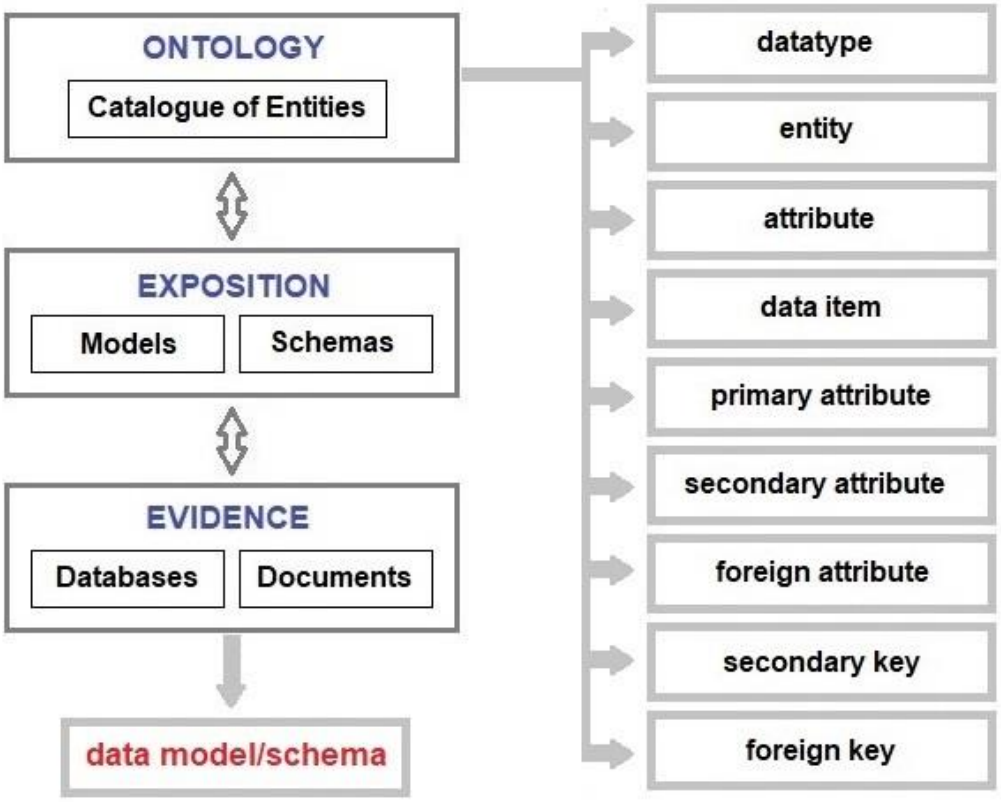
Supply of Personal Protective Equipment for Healthcare W...

DEPARTMENT OF HEALTH AND SOCIAL CARE

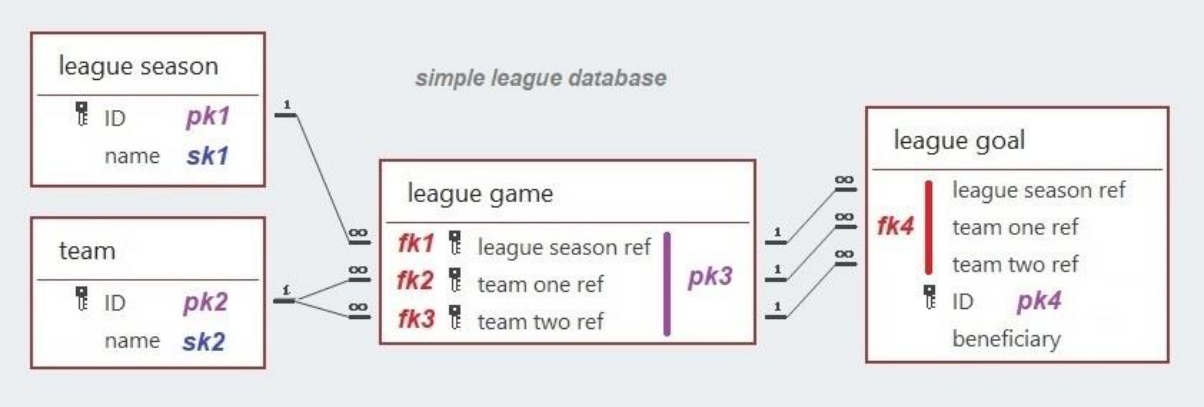
The Department of Health and Social Care (the "Authority") has awarded a contract to PPE **Medpro** Limited for the supply of Personal Protective Equipment (gowns) to protect Health and Care workers from contracting the Covid 19 disease.

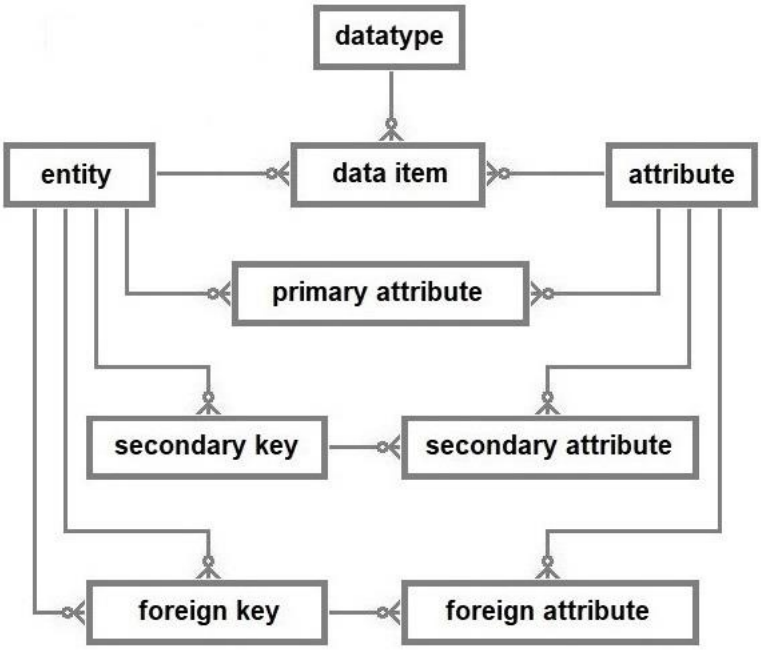
Procurement stage Awarded contract
Contract location United Kingdom
Awarded value £122,000,000
Awarded supplier PPE **Medpro** Limited
Publication date 7 September 2020, last edited 5 October 2020

Chapter 13 – Anatomy of a Database



This database model comprises the nine constructs listed above. They are the entities used when designing the databases discussed previously. The *simple league* database will be deconstructed to explain the database model.





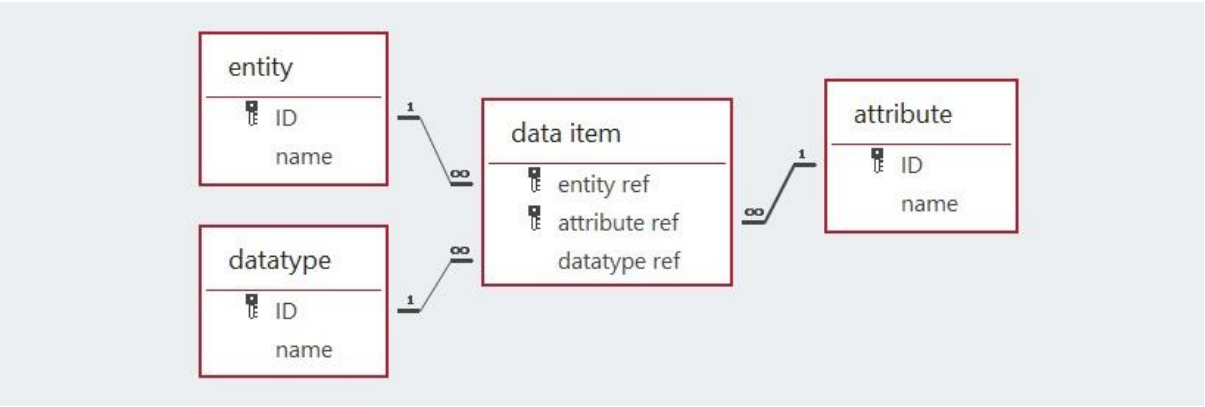
In the *simple league* database, the only datatype used was the text string and that applies to all twelve data items. The twelve data items are distributed between the four entity types. The *beneficiary* attribute of the *league goal* entity type has a specialised restricted string datatype that only provides two options, *team one* and *team two*. Unlike the *beneficiary* attribute, the other five attributes used to specify the data items are common to more than one entity type, e.g., *ID* and *name*.

datatype	string

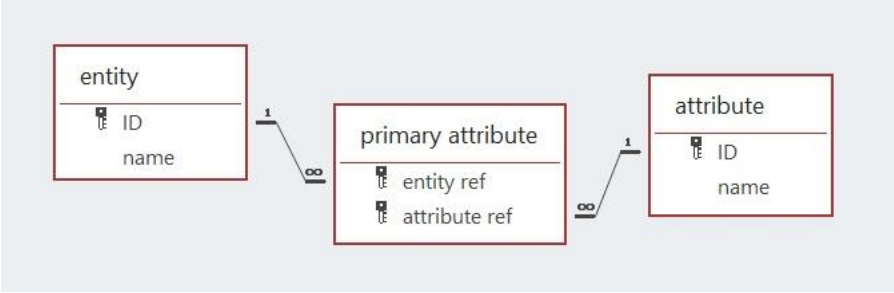
entity
league season
team
league game
league goal

attribute
ID
name
league season ref
team one ref
team two ref
beneficiary

data item
league season/ID: string
league season/name: string
team/ID: string
team/name: string
league game/league season ref: string
league game/team one ref: string
league game/team two ref: string
league goal/league season ref: string
league goal/team one ref: string
league goal/team two ref: string
league goal/ID: string
league goal/beneficiary: string

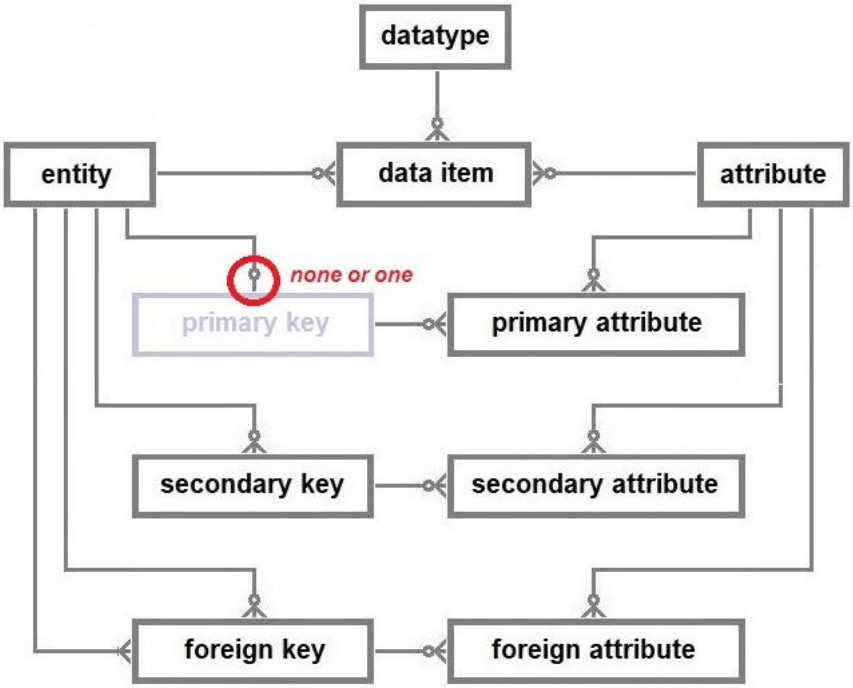


The data model of a database above does not feature a primary key. This is because all primary attributes are linked to an entity and knowing that an entity can only have one primary key allows the existence and composition of the primary key to be deduced from the designated primary attributes.



However, the real world intrudes, necessitating the explicit statement of the primary key's existence if there is one. Data management systems seem to require evidence that a primary key has been designated. Therefore, to allow the database management system to reference the primary attribute(s), a single unique label is used to signpost the primary attribute(s). As a result of that real world requirement, primary keys have been named in the data models discussed above. MS Access automatically assigns the label "PrimaryKey" to the designated primary attribute(s) of a table. An XML schema also requires primary keys to be named *(see chapter 5)*.

The data model below therefore includes a ghost primary key, but it is only a sop to real world data management techniques and adds nothing to the understanding of a database's structure.

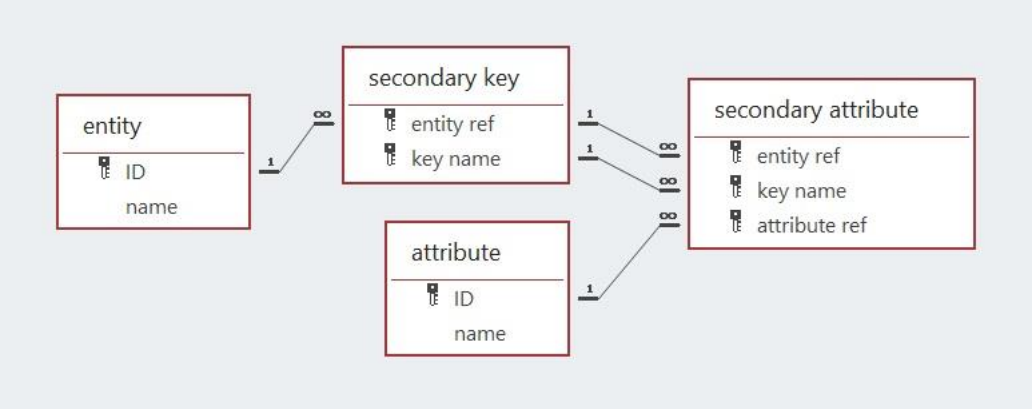


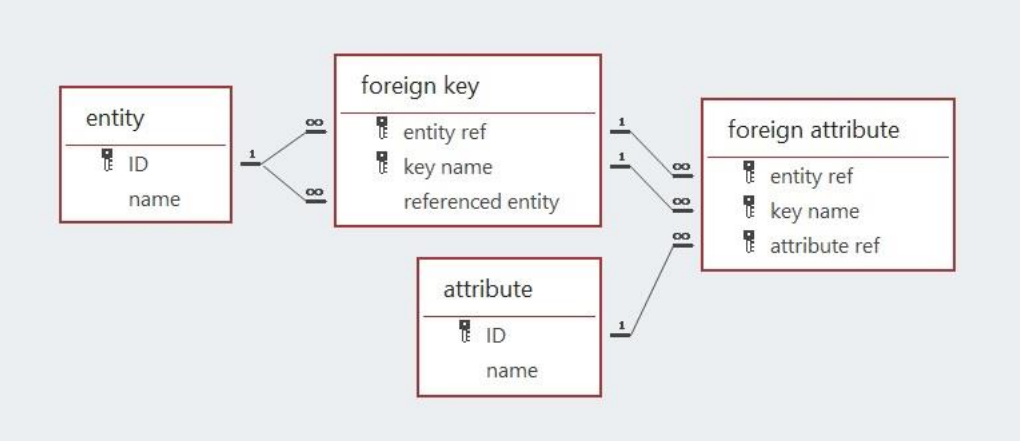
primary key	primary key 1	league season/ID *(pk1)*	*primary attribute*
	primary key 2	team/ID *(pk2)*	
	primary key 3	league game/league season ref *(pk3)*	
		league game/team one ref *(pk3)*	
		league game/team two ref *(pk3)*	
	primary key 4	league goal/ID *(pk4)*	

An entity may have multiple secondary and foreign keys. This luxury generates the need to differentiate these keys by giving them unique names. Note that the names of secondary and foreign keys cannot be changed because they are part of multi-attribute primary keys.

secondary key	secondary key 1	league season/name *(sk1)*	*secondary attribute*
	secondary key 2	team/name *(sk2)*	

foreign key	foreign key 1	league game/league season ref *(fk1)*	*foreign attribute*
	foreign key 2	league game/team one ref *(fk2)*	
	foreign key 3	league game/team two ref *(fk3)*	
	foreign key 4	league goal/league season ref *(fk4)*	
		league goal/team one ref *(fk4)*	
		league goal/team two ref *(fk4)*	



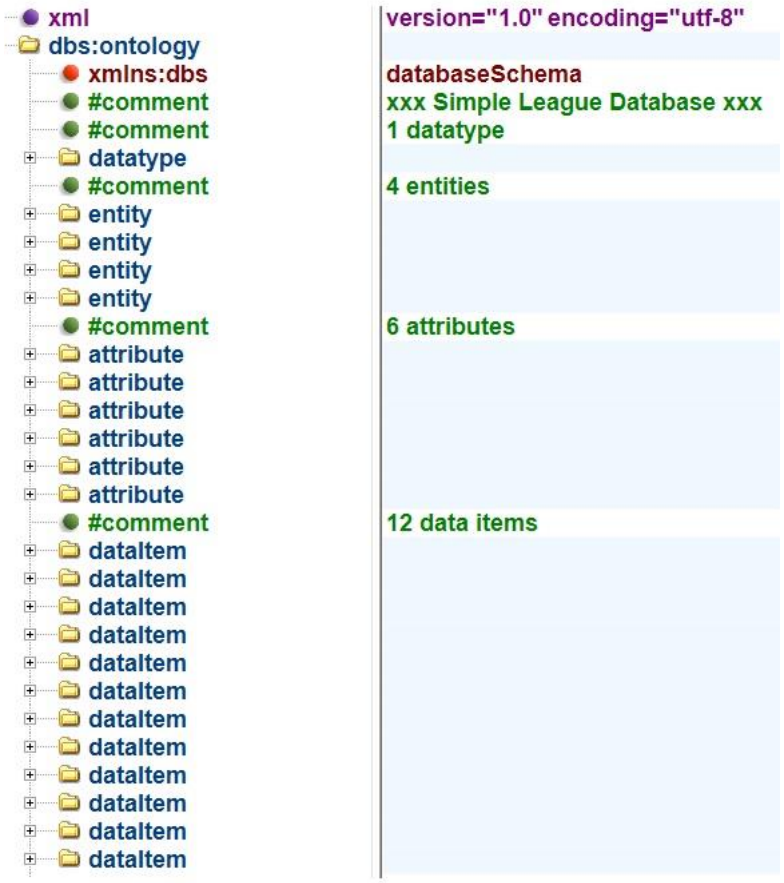


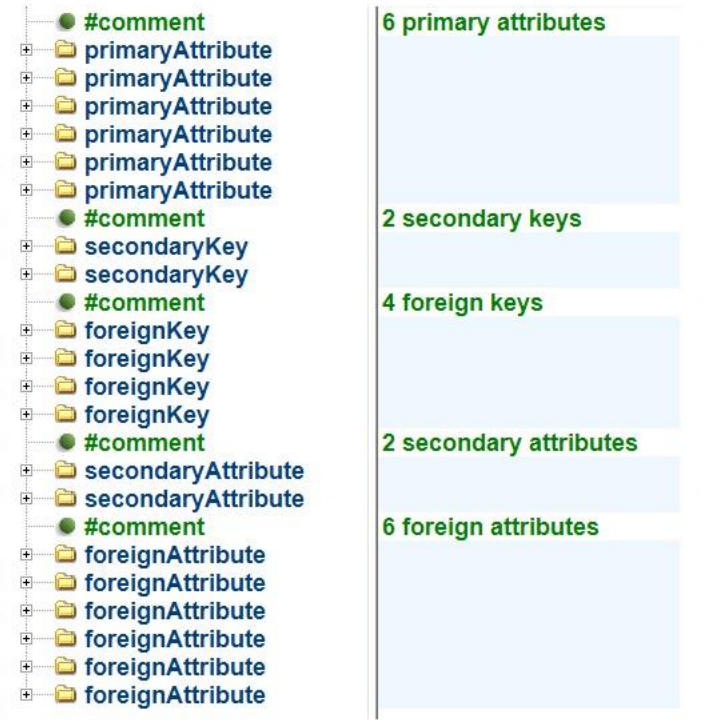
The above explanation should be consistent with the design of the databases discussed earlier. My apologies if that is unclear. Well, if it is clear, you should now have an appreciation of how digital databases may be built with the use of three types of keys to structure the data items being collected.

The appendix, unfortunately interminable, is an indulgence on my part and is unlikely to be of any significant use to most readers, but it may help diligent students. It provides an XML document *(instance)* with the specifications of the *simple league database* structure and an XML schema for the database model discussed in this chapter. The schema was validated using the Corefilling utility referenced in Chapter 5,

Appendix – Database Instance and Schema

The following XML document is a machine-readable version of the *simple league season* database structure introduced in Chapter 2.






```
<dbs:ontology>
  <!--xxx Simple League Database xxx -->

  <!--1 datatype -->
  <datatype>
     <ID>df%3@5</ID>
     <name>string</name>
   </datatype>

  <!--4 entities -->
  <entity>
     <ID>7£o0#2</ID>
     <name>league season</name>
   </entity>
  <entity>
     <ID>r$0gT7</ID>
     <name>team</name>
   </entity>
  <entity>
     <ID>b3%w?r</ID>
     <name>league game</name>
   </entity>
  <entity>
     <ID>m3%oB8</ID>
     <name>league goal</name>
   </entity>
```

```
<!--6 attributes -->
<attribute>
   <ID>v&9s[3</ID>
   <name>ID</name>
 </attribute>
<attribute>
   <ID>5m?#2d</ID>
   <name>name</name>
 </attribute>
<attribute>
   <ID>o1@G£9</ID>
   <name>league season ref</name>
 </attribute>
<attribute>
   <ID>q2+M9£</ID>
   <name>team one ref</name>
 </attribute>
<attribute>
   <ID>fL0@S8</ID>
   <name>team two ref</name>
 </attribute>
<attribute>
   <ID>cd4$&4</ID>
   <name>beneficiary</name>
 </attribute>
```

```
<!--12 data items -->
<dataItem>
   <entityRef>7£o0#2</entityRef>
   <attributeRef>v&9s[3</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>7£o0#2</entityRef>
   <attributeRef>5m?#2d</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>r$0gT7</entityRef>
   <attributeRef>v&9s[3</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>r$0gT7</entityRef>
   <attributeRef>5m?#2d</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>b3%w?r</entityRef>
   <attributeRef>o1@G£9</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>b3%w?r</entityRef>
   <attributeRef>q2+M9£</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
```

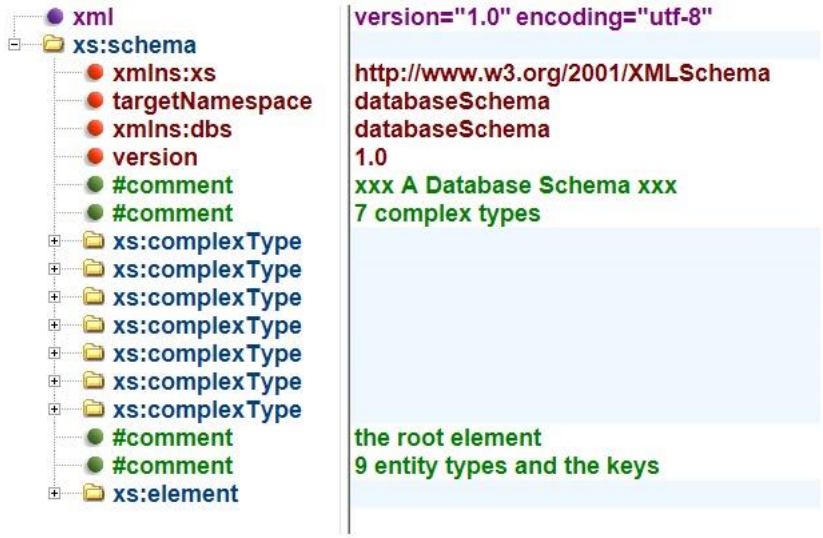
```
<dataItem>
   <entityRef>b3%w?r</entityRef>
   <attributeRef>fL0@S8</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>m3%oB8</entityRef>
   <attributeRef>o1@G£9</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>m3%oB8</entityRef>
   <attributeRef>q2+M9£</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>m3%oB8</entityRef>
   <attributeRef>fL0@S8</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>m3%oB8</entityRef>
   <attributeRef>v&9s[3</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
<dataItem>
   <entityRef>m3%oB8</entityRef>
   <attributeRef>cd4$&4</attributeRef>
   <datatypeRef>df%3@5</datatypeRef>
 </dataItem>
```

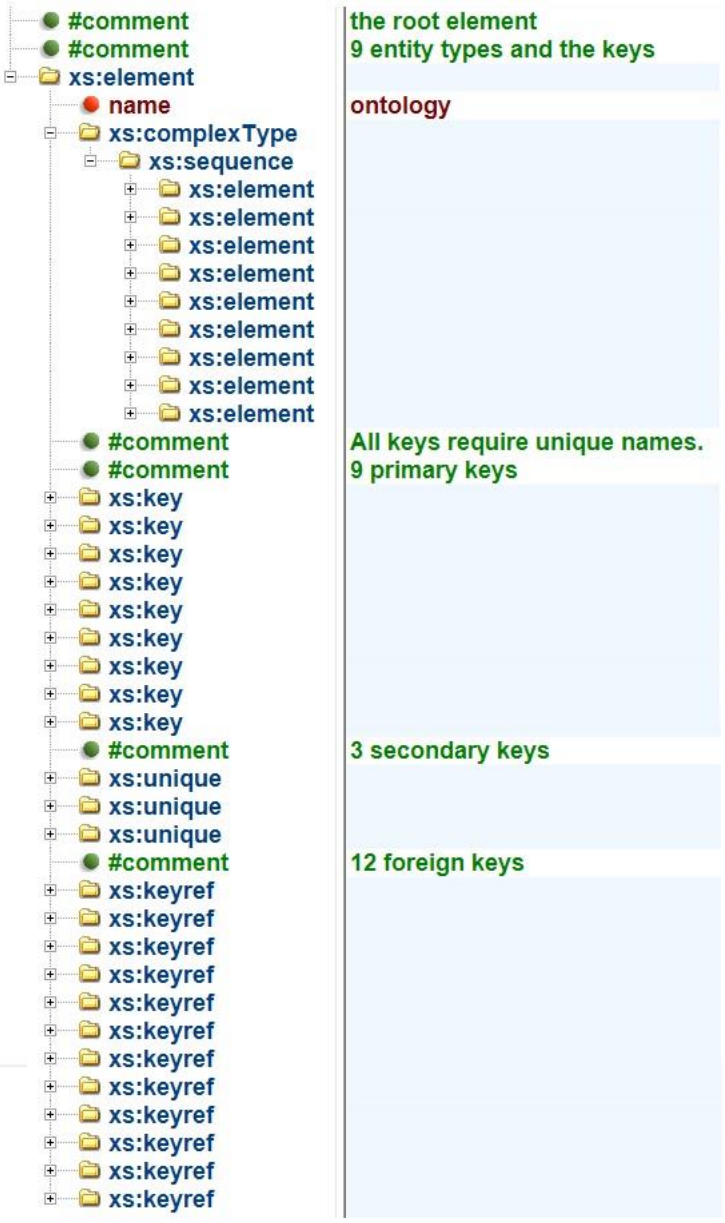
```
<!--6 primary attributes -->
<primaryAttribute>
   <entityRef>7£o0#2</entityRef>
   <attributeRef>v&9s[3</attributeRef>
 </primaryAttribute>
<primaryAttribute>
   <entityRef>r$0gT7</entityRef>
   <attributeRef>v&9s[3</attributeRef>
 </primaryAttribute>
<primaryAttribute>
   <entityRef>b3%w?r</entityRef>
   <attributeRef>o1@G£9</attributeRef>
 </primaryAttribute>
<primaryAttribute>
   <entityRef>b3%w?r</entityRef>
   <attributeRef>q2+M9£</attributeRef>
 </primaryAttribute>
<primaryAttribute>
   <entityRef>b3%w?r</entityRef>
   <attributeRef>fL0@S8</attributeRef>
 </primaryAttribute>
<primaryAttribute>
   <entityRef>m3%oB8</entityRef>
   <attributeRef>v&9s[3</attributeRef>
 </primaryAttribute>
```

```
<!--2 secondary keys -->
<secondaryKey>
   <entityRef>7£o0#2</entityRef>
   <keyName>secondary key 1</keyName>
 </secondaryKey>
<secondaryKey>
   <entityRef>r$0gT7</entityRef>
   <keyName>secondary key 2</keyName>
 </secondaryKey>

<!--4 foreign keys -->
<foreignKey>
   <entityRef>b3%w?r</entityRef>
   <keyName>foreign key 1</keyName>
   <referencedEntity>7£o0#2</referencedEntity>
 </foreignKey>
<foreignKey>
   <entityRef>b3%w?r</entityRef>
   <keyName>foreign key 2</keyName>
   <referencedEntity>r$0gT7</referencedEntity>
 </foreignKey>
<foreignKey>
   <entityRef>b3%w?r</entityRef>
   <keyName>foreign key 3</keyName>
   <referencedEntity>r$0gT7</referencedEntity>
 </foreignKey>
<foreignKey>
   <entityRef>m3%oB8</entityRef>
   <keyName>foreign key 4</keyName>
   <referencedEntity>b3%w?r</referencedEntity>
 </foreignKey>

<!--2 secondary attributes -->
<secondaryAttribute>
   <entityRef>7£o0#2</entityRef>
   <keyName>secondary key 1</keyName>
   <attributeRef>5m?#2d</attributeRef>
 </secondaryAttribute>
<secondaryAttribute>
   <entityRef>r$0gT7</entityRef>
   <keyName>secondary key 2</keyName>
   <attributeRef>5m?#2d</attributeRef>
 </secondaryAttribute>
```

```
<!--6 foreign attributes -->
<foreignAttribute>
   <entityRef>b3%w?r</entityRef>
   <keyName>foreign key 1</keyName>
   <attributeRef>o1@G£9</attributeRef>
 </foreignAttribute>
<foreignAttribute>
   <entityRef>b3%w?r</entityRef>
   <keyName>foreign key 2</keyName>
   <attributeRef>q2+M9£</attributeRef>
 </foreignAttribute>
<foreignAttribute>
   <entityRef>b3%w?r</entityRef>
   <keyName>foreign key 3</keyName>
   <attributeRef>fL0@S8</attributeRef>
 </foreignAttribute>
<foreignAttribute>
   <entityRef>m3%oB8</entityRef>
   <keyName>foreign key 4</keyName>
   <attributeRef>o1@G£9</attributeRef>
 </foreignAttribute>
<foreignAttribute>
   <entityRef>m3%oB8</entityRef>
   <keyName>foreign key 4</keyName>
   <attributeRef>q2+M9£</attributeRef>
 </foreignAttribute>
<foreignAttribute>
   <entityRef>m3%oB8</entityRef>
   <keyName>foreign key 4</keyName>
   <attributeRef>fL0@S8</attributeRef>
 </foreignAttribute>
</dbs:ontology>
```

The schema necessary to verify that the instance/XML document above conforms to the database model discussed in the previous chapter is provided below. The outline of the schema is shown prior to the detailed schema being set out.


xml
version="1.0" encoding="utf-8"
xs:schema
xmlns:xs
http://www.w3.org/2001/XMLSchema
targetNamespace
databaseSchema
xmlns:dbs
databaseSchema
version
1.0
#comment
xxx A Database Schema xxx
#comment
7 complex types
xs:complexType
xs:complexType
xs:complexType
xs:complexType
xs:complexType
xs:complexType
xs:complexType
#comment
the root element
#comment
9 entity types and the keys
xs:element


#comment
the root element
#comment
9 entity types and the keys
xs:element
name
ontology
xs:complexType
xs:sequence
xs:element
xs:element
xs:element
xs:element
xs:element
xs:element
xs:element
xs:element
xs:element
#comment
All keys require unique names.
#comment
9 primary keys
xs:key
xs:key
xs:key
xs:key
xs:key
xs:key
xs:key
xs:key
xs:key
#comment
3 secondary keys
xs:unique
xs:unique
xs:unique
#comment
12 foreign keys
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref
xs:keyref

```
<xs:schema targetNamespace="databaseSchema" version="1.0">
  <!--xxx A Database Schema xxx -->
  <!--7 complex types -->
  <xs:complexType name="namedEntity">
    <xs:sequence>
      <xs:element name="ID" type="xs:string"/>
      <xs:element name="name" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="entityReference">
    <xs:sequence>
      <xs:element name="entityRef" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="entityAttributeCombo">
    <xs:complexContent>
      <xs:extension base="dbs:entityReference">
        <xs:sequence>
          <xs:element name="attributeRef" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="dataItemDefn">
    <xs:complexContent>
      <xs:extension base="dbs:entityAttributeCombo">
        <xs:sequence>
          <xs:element name="datatypeRef" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
<xs:complexType name="keyNameDefn">
  <xs:complexContent>
    <xs:extension base="dbs:entityReference">
      <xs:sequence>
        <xs:element name="keyName" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="foreignKeyDefn">
  <xs:complexContent>
    <xs:extension base="dbs:keyNameDefn">
      <xs:sequence>
        <xs:element name="referencedEntity" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="nonPrimeAttribute">
  <xs:complexContent>
    <xs:extension base="dbs:keyNameDefn">
      <xs:sequence>
        <xs:element name="attributeRef" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!--the root element -->
<!--9 entity types and the keys -->
<xs:element name="ontology">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="datatype" type="dbs:namedEntity"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="entity" type="dbs:namedEntity"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="attribute" type="dbs:namedEntity"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="dataItem" type="dbs:dataItemDefn"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="primaryAttribute"
        type="dbs:entityAttributeCombo" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="secondaryKey" type="dbs:keyNameDefn"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="foreignKey" type="dbs:foreignKeyDefn"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="secondaryAttribute"
        type="dbs:nonPrimeAttribute" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="foreignAttribute"
        type="dbs:nonPrimeAttribute" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

```
<!--All keys require unique names. -->
<!--9 primary keys -->
<xs:key name="primaryKey1">
   <xs:selector xpath="entity"/>
   <xs:field xpath="ID"/>
 </xs:key>
<xs:key name="primaryKey2">
   <xs:selector xpath="attribute"/>
   <xs:field xpath="ID"/>
 </xs:key>
<xs:key name="primaryKey3">
   <xs:selector xpath="datatype"/>
   <xs:field xpath="ID"/>
 </xs:key>
<xs:key name="primaryKey4">
   <xs:selector xpath="dataItem"/>
   <xs:field xpath="entityRef"/>
   <xs:field xpath="attributeRef"/>
 </xs:key>
<xs:key name="primaryKey5">
   <xs:selector xpath="primaryAttribute"/>
   <xs:field xpath="entityRef"/>
   <xs:field xpath="attributeRef"/>
 </xs:key>
<xs:key name="primaryKey6">
   <xs:selector xpath="secondaryKey"/>
   <xs:field xpath="entityRef"/>
   <xs:field xpath="keyName"/>
 </xs:key>
```

```
<xs:key name="primaryKey7">
   <xs:selector xpath="foreignKey"/>
   <xs:field xpath="entityRef"/>
   <xs:field xpath="keyName"/>
 </xs:key>
<xs:key name="primaryKey8">
   <xs:selector xpath="secondaryAttribute"/>
   <xs:field xpath="entityRef"/>
   <xs:field xpath="keyName"/>
   <xs:field xpath="attributeRef"/>
 </xs:key>
<xs:key name="primaryKey9">
   <xs:selector xpath="foreignAttribute"/>
   <xs:field xpath="entityRef"/>
   <xs:field xpath="keyName"/>
   <xs:field xpath="attributeRef"/>
 </xs:key>
<!--3 secondary keys -->
<xs:unique name="secondaryKey1">
   <xs:selector xpath="entity"/>
   <xs:field xpath="name"/>
 </xs:unique>
<xs:unique name="secondaryKey2">
   <xs:selector xpath="attribute"/>
   <xs:field xpath="name"/>
 </xs:unique>
<xs:unique name="secondaryKey3">
   <xs:selector xpath="datatype"/>
   <xs:field xpath="name"/>
 </xs:unique>
```

```
<!--12 foreign keys -->
<xs:keyref name="foreignKey1" refer="dbs:primaryKey1">
   <xs:selector xpath="dataItem"/>
   <xs:field xpath="entityRef"/>
 </xs:keyref>
<xs:keyref name="foreignKey2" refer="dbs:primaryKey2">
   <xs:selector xpath="dataItem"/>
   <xs:field xpath="attributeRef"/>
 </xs:keyref>
<xs:keyref name="foreignKey3" refer="dbs:primaryKey3">
   <xs:selector xpath="dataItem"/>
   <xs:field xpath="datatypeRef"/>
 </xs:keyref>
<xs:keyref name="foreignKey4" refer="dbs:primaryKey1">
   <xs:selector xpath="primaryAttribute"/>
   <xs:field xpath="entityRef"/>
 </xs:keyref>
<xs:keyref name="foreignKey5" refer="dbs:primaryKey2">
   <xs:selector xpath="primaryAttribute"/>
   <xs:field xpath="attributeRef"/>
 </xs:keyref>
<xs:keyref name="foreignKey6" refer="dbs:primaryKey1">
   <xs:selector xpath="secondaryKey"/>
   <xs:field xpath="entityRef"/>
 </xs:keyref>
```

```
    <xs:keyref name="foreignKey7" refer="dbs:primaryKey6">
       <xs:selector xpath="secondaryAttribute"/>
       <xs:field xpath="entityRef"/>
       <xs:field xpath="keyName"/>
     </xs:keyref>
    <xs:keyref name="foreignKey8" refer="dbs:primaryKey2">
       <xs:selector xpath="secondaryAttribute"/>
       <xs:field xpath="attributeRef"/>
     </xs:keyref>
    <xs:keyref name="foreignKey9" refer="dbs:primaryKey1">
       <xs:selector xpath="foreignKey"/>
       <xs:field xpath="entityRef"/>
     </xs:keyref>
    <xs:keyref name="foreignKey10" refer="dbs:primaryKey1">
       <xs:selector xpath="foreignKey"/>
       <xs:field xpath="referencedEntity"/>
     </xs:keyref>
    <xs:keyref name="foreignKey11" refer="dbs:primaryKey7">
       <xs:selector xpath="foreignAttribute"/>
       <xs:field xpath="entityRef"/>
       <xs:field xpath="keyName"/>
     </xs:keyref>
    <xs:keyref name="foreignKey12" refer="dbs:primaryKey2">
       <xs:selector xpath="foreignAttribute"/>
       <xs:field xpath="attributeRef"/>
     </xs:keyref>
   </xs:element>
</xs:schema>
```